Étude de l'hybridation des méta-heuristiques, application à un problème d'ordonnancement de type jobshop

David Duvivier

Côte-d'Opale, Calais
Laboratoire d'Informatique du Littoral

N° d'ordre : Année : 2000

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DU LITTORAL
CÔTE D'OPALE

Discipline : Informatique

présentée et soutenue publiquement

par David DUVIVIER

le 12 décembre 2000

Titre :

Étude de l'hybridation des méta-heuristiques, application à un problème d'ordonnancement de type jobshop.

Directeur de thèse : Philippe PREUX

JURY

H. BASSON, Professeur à l'Université du Littoral Côte d'Opale ..... Président
A. ARTIBA, Professeur aux Facultés Universitaires Catholiques de Mons ..... Rapporteur
G. VENTURINI, Professeur à l'Université de Tours ..... Rapporteur
A. FRÉVILLE, Professeur à l'Université de Valenciennes et du Hainaut Cambrésis Examinateur
E.G. TALBI, Maître de conférences à l'Université de Lille 1, HDR ..... Examinateur
Ph. PREUX, Professeur à l'Université du Littoral Côte d'Opale ..... Directeur de thèse

Á ceux qui sont partis trop tôt,

Á Maryline, ma femme,

Á Alexandre, notre fils ...

Remerciements

Le travail présenté dans ce mémoire a été initié au laboratoire d'informatique fondamentale de Lille (LIFL). *Il s'est poursuivi au laboratoire d'informatique du Littoral* (LIL) *dont je tiens à remercier le directeur, Monsieur le Professeur Philippe* Preux, *pour m'avoir accueilli au sein du* LIL.

Mes remerciements vont ensuite aux membres du Jury qui ont accepté de consacrer une partie de leur temps à l'évaluation de mes travaux :

À M. Henri Basson, *Professeur à l'Université du Littoral Côte d'Opale, tout d'abord pour avoir accepté de présider mon jury de thèse, mais aussi pour les nombreuses discussions constructives et amicales que nous avons eues au sein du* LIL *...*

À M. Abdelhakim Artiba, *Professeur aux Facultés Universitaires Catholiques de Mons, non seulement pour avoir accepté d'être rapporteur ainsi que pour ses précieux conseils, mais également pour m'avoir accueilli dans son équipe jeune et dynamique au sein du centre de recherches et d'études en gestion industrielle (CREGI) ;*

À M. Gilles Venturini, *Professeur à l'Université de Tours, tout d'abord pour avoir accepté d'être rapporteur, mais aussi pour ses nombreuses remarques et critiques constructives sur mes travaux qui ont largement contribuées à l'amélioration de ce manuscrit ;*

À M. Arnaud Fréville, *Professeur à l'Université de Valenciennes et du Hainaut Cambrésis, pour avoir accepté de participer à mon jury de thèse ainsi que pour les discussions constructives lors de la soutenance ;*

À M. El-Ghazali Talbi, *Maître de conférences à l'Université de Lille 1, non seulement pour sa participation à mon jury de thèse, mais également pour avoir collaboré activement à ce travail dans le cadre du groupe de recherche* PERFORM *;*

Mes remerciements vont de nouveau à M. Philippe Preux, *Professeur à l'Université du Littoral Côte d'Opale, non seulement pour la confiance qu'il m'a accordée en tant que directeur de thèse, mais aussi pour le soutien qu'il m'a apporté au fil de mes travaux.*

Je tiens également à remercier les membres du LIL *et de l'IUT du Littoral qui ont contribué à créer un cadre de travail agréable tout au long de la réalisation de cette thèse. J'ai tout particulièrement apprécié les échanges fructueux au sein de l'équipe* MESC *et du groupe de recherche* PERFORM. *Ils ont largement participé au développement de ce travail. Enfin, je remercie les membres du* CREGI *au sein duquel je poursuis mes recherches.*

Mes derniers mots seront pour ma famille : mes parents qui ont toujours été présents pour me soutenir et qui s'en sont allés trop tôt pour assister à l'aboutissement de ce travail ... Pour ma belle famille qui a su m'épauler dans les moments difficiles. Et tout naturellement pour ma femme, Maryline, *qui a su faire preuve de beaucoup de patience tout au long de la réalisation de cette thèse et m'a apporté, au travers de notre fils* Alexandre, *la plus forte des motivations pour aller au bout de ce travail. Pardon par avance à ceux que je ne cite pas explicitement afin de garder une taille raisonnable pour ces remerciements. A tous, je dédie ce mémoire qui n'aurait jamais vu le jour sans vous.*

12 mars 2001

Table des matières

Chapitre 1

Introduction générale

La plupart des problèmes industriels sont d'une telle « complexité » que le nombre de solutions potentielles croît exponentiellement avec la taille du problème. Les méthodes exactes deviennent rapidement inutilisables pour des instances de grande taille. Il s'avère alors nécessaire d'utiliser des méthodes heuristiques pour les résoudre en un temps raisonnable. Les méthodes heuristiques offrent l'avantage de ne parcourir qu'une faible fraction de l'espace de recherche pour parvenir à une solution acceptable. Dans le cadre de notre étude, nous nous intéressons à des heuristiques générales applicables à n'importe quel problème : les méta-heuristiques.

Chaque méthode de résolution, approchée ou exacte, possède des propriétés distinctes. Les algorithmes hybrides allient plusieurs d'entre-elles afin d'obtenir des méthodes plus performantes et plus robustes. La robustesse caractérise la reproductibilité des résultats obtenus. La performance peut être définie en terme de qualité de solution, de durée d'exécution, de nombre d'appels à la fonction coût, ou encore de taille mémoire occupée ... Le plus souvent il s'agit de trouver un compromis « acceptable » entre plusieurs de ces critères.

L'objectif essentiel de nos travaux concerne l'utilisation de modèles hybrides basés sur la collaboration de plusieurs méthodes de résolution : bien qu'elle ne soit pas une notion nouvelle dans le monde de la recherche opérationnelle, il s'agit d'une technique récemment apparue dans le domaine des algorithmes évolutifs. Elle permet d'obtenir d'excellentes performances en tirant au mieux profit des caractéristiques des algorithmes évolutifs d'une part, et des méthodes de recherche intégrées dans les algorithmes hybrides d'autre part. Nos travaux visent à dégager des résultats concernant l'utilisation des modèles hybrides dans le cadre de l'optimisation combinatoire. Plus que les performances elles-mêmes, nous nous intéressons tout particulièrement à la compréhension du fonctionnement des méthodes de résolution ainsi qu'à l'analyse de l'influence de la coopération de plusieurs méthodes de recherche sur la qualité des solutions engendrées.

Nous concentrons nos travaux sur les méthodes amélioratrices : ces méthodes de recherche sont basées sur un opérateur de déplacement chargé du cheminement dans l'espace de recherche en passant de « proche en proche », d'une solution à une autre. Le codage des solutions, ainsi que l'opérateur utilisé conditionnent en grande partie les performances des méthodes de recherche. C'est pourquoi, le second point abordé est le choix ou la définition d'un couple $\langle \text{codage}, \text{opérateur de déplacement} \rangle$ performant. Nous utilisons la notion de paysage adaptatif pour guider notre choix parmi les différentes possibilités qui s'offrent à nous.

Enfin, le dernier point abordé porte sur l'évaluation des performances des heuristiques. Nous avons choisi d'évaluer ces heuristiques sur la résolution d'un problème d'ordonnancement de type *jobshop* (JSP). Il s'agit d'un problème $\mathcal{NP}$-dur [GJ79], très difficile à résoudre, et d'une grande

utilité pratique. Les techniques et outils mis en œuvre pour la résolution du *jobshop* peuvent être appliqués directement ou indirectement à un grand nombre de problèmes voisins, ce dans beaucoup de domaines (problèmes d'atelier, conception d'emploi du temps, planification des décollages et atterrissages d'avions, ordonnancement d'instructions (micro-code), constitution et ordonnancement de trains, construction de cartes génétiques, routage de paquets dans un réseau ...). Le cheminement des heuristiques est étudié dans plusieurs espaces de recherche engendrés par différents couples $\langle$codage, opérateur$\rangle$ lors de la résolution du *jobshop*. Nous étudions tout particulièrement le comportement des heuristiques par rapport à la présence de plateaux dans l'espace de recherche. L'utilisation de fonctions multicritères et d'un voisinage restreint nous permet de modifier cet espace afin d'améliorer les performances.

Nous organisons notre propos en quatre parties principales :

La première partie, est un état de l'art des techniques de résolutions, axé sur les méta-heuristiques. Une présentation des méthodes hybrides termine cette première partie.

Ensuite, dans la seconde partie, nous décrivons le problème traité et les principales méthodes de résolution utilisées.

La troisième partie est une présentation de nos travaux.

La dernière partie est constituée des annexes, de la bibliographie, et des listes destinées à faciliter la recherche d'informations dans ce document.

Partie I

État de l'art

Chapitre 2

Méthodes de résolution

La résolution informatique de tout problème passe par l'utilisation d'une méthode de résolution.

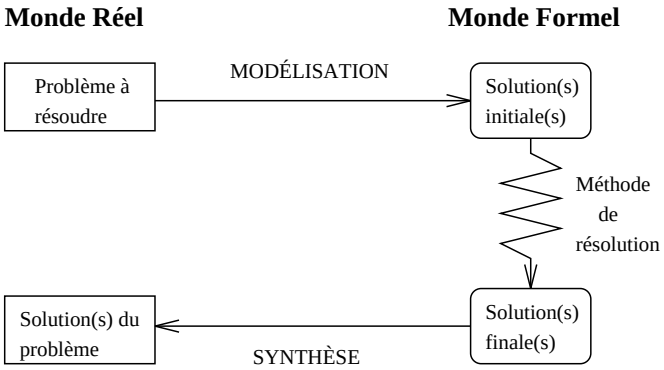


Figure 2.1: Modélisation / synthèse – la résolution informatique d'un problème se décompose en deux phases principales : la modélisation au cours de laquelle le problème issu du « monde réel » est transcrit en une représentation informatique, et la synthèse des résultats qui consiste à exprimer dans le monde réel la solution obtenue par résolution informatique.

Pour résoudre un problème à partir d'une méthode de résolution, il faut définir une représentation d'une solution sous forme d'une structure de données particulière par l'établissement d'un *codage* et des *opérateurs* associés : c'est la phase de *modélisation* (voir figure 2.1). La phase suivante consiste à exécuter la méthode de résolution. La dernière phase synthétise les résultats issus de la méthode de résolution et les retranscrit dans le monde réel (*i.e.* le problème posé) : c'est la *synthèse* des résultats.

Dans ce chapitre, nous présentons différentes méthodes de résolution ainsi que plusieurs critères de comparaison. L'ensemble de ces méthodes est tellement vaste qu'il est impossible de toutes les exposer. Aussi, dans le cadre de notre étude, nous présentons les principales méthodes de résolution.

2.1 Optimisation combinatoire

Avant même de commencer notre étude, il convient de préciser le cadre dans lequel nous utilisons les méthodes de résolution : nous traitons des problèmes *d'optimisation combinatoire*.

Définition 1 (solution réalisable)
Soit un problème Λ défini par un ensemble de contraintes, une solution est dite réalisable si elle satisfait toutes les contraintes de Λ.

Définition 2 (solution non réalisable)
Soit un problème Λ défini par un ensemble de contraintes, une solution est dite non réalisable, ou invalide, si elle ne satisfait pas toutes les contraintes de Λ.

Définition 3 (optimisation combinatoire)
L'optimisation combinatoire est le domaine des mathématiques discrètes qui traite de la résolution du problème Λ suivant : soient $\mathcal{E}$ un ensemble fini de solutions réalisables et f une fonction permettant d'évaluer chaque solution réalisable, il s'agit de déterminer une solution $s^* \in \mathcal{E}$ qui optimise f, c'est-à-dire telle que $f(s^*) = \min_{s \in \mathcal{E}} f(s)$ *ou* $f(s^*) = \max_{s \in \mathcal{E}} f(s)$

L'ensemble $\mathcal{E}$ est en général défini par un ensemble C de contraintes matérialisées par des équations ou des inéquations délimitant les frontières de $\mathcal{E}$. La fonction f est appelée la fonction coût.

À l'instar de [ERR94], nous divisons les problèmes de recherche en trois catégories : les problèmes de satisfaction de contraintes, les problèmes d'optimisation sans contrainte, et les problèmes d'optimisation avec contraintes.

Les problèmes de satisfaction de contraintes, consistent à trouver une instanciation d'un ensemble de variables qui vérifie l'ensemble des contraintes définies sur ces variables.

Pour les problèmes d'optimisation sans contrainte, l'ensemble C est vide et chaque point de l'espace est une solution réalisable. Certains problèmes comportant des contraintes peuvent être représentés sous cette forme. Dans ce cas, les contraintes sont reportées dans la fonction coût via un mécanisme de pénalité qui tend à « décourager » l'apparition de solutions qui ne respectent pas les contraintes imposées par le problème.

La forme la plus générale correspond à un regroupement de ces deux catégories de problèmes : il s'agit des problèmes d'optimisation avec contraintes. Tout problème d'optimisation peut se mettre sous cette forme. L'objectif consiste à optimiser la fonction coût en respectant les contraintes ce qui correspond tout à fait à la définition 3.

Dans le cadre de nos travaux, nous nous limitons à cette dernière catégorie car elle correspond au cas le plus général, et plus particulièrement au problème choisi pour appliquer les techniques exposées. Nous allons maintenant aborder les principales méthodes de résolution.

2.2 Principales méthodes de résolution

En première instance, les méthodes de résolution peuvent être réparties en deux principales familles : les méthodes de résolution dédiées et les méthodes de résolution générales, ou méta-heuristiques.

Comme leur nom le suggère, les méthodes dédiées sont conçues pour résoudre un problème particulier. Elles sont extrêmement efficaces pour résoudre le problème pour lequel elles ont été conçues, mais inapplicables à d'autres problèmes (ou elles le sont au prix de performances médiocres).

En revanche, les méta-heuristiques sont des méthodes applicables à un très grand nombre de problèmes, mais sont susceptibles d'offrir des performances inférieures aux méthodes dédiées.

Dans la suite de ce mémoire, nous verrons que la frontière entre ces deux familles n'est pas hermétique, et les méthodes les plus performantes à ce jour intègrent ces deux types de méthodes au sein de *méthodes hybrides*.

Avant d'aborder l'étude des principales méta-heuristiques dans un prochain chapitre, nous présentons une classification des méthodes de résolution afin de situer les méta-heuristiques parmi ces méthodes. Cette classification ne prétend pas être exhaustive, mais a pour objectif de faire ressortir les principales caractéristiques des méthodes de résolution.

Les méthodes de résolution peuvent être réparties en quatre catégories :

- méthodes exactes ;
- méthodes de relaxation ;
- méthodes heuristiques ;
- méthodes hybrides.

Les méthodes *exactes* fournissent systématiquement une solution (optimale) au problème traité si une telle solution existe. Dans le cas contraire, ce type de méthode permet d'affirmer qu'il n'existe pas de solution au problème traité.

Les méthodes *de relaxation* fournissent une solution approchée au problème traité. Elle sont en général conçues de manière à ce que la solution obtenue puisse être située (en terme de coût) par rapport à la valeur optimale : de telles méthodes permettent d'obtenir des bornes inférieures ou supérieures de la valeur optimale du coût.

Les méthodes *heuristiques* permettent d'obtenir une solution acceptable[1] au problème en un temps raisonnable. Malheureusement il n'est en général pas possible de garantir la qualité de la solution obtenue (en terme de coût).

Les méthodes *hybrides* sont constituées de plusieurs méthodes issues des catégories sus-citées, ou de méthodes particulières n'entrant pas dans ces catégories. Nous reviendrons dans un prochain chapitre sur les méthodes hybrides.

Ces catégories sont loin d'exprimer toutes les particularités des méthodes de recherche. Ainsi, parmi ces méthodes, certaines sont déterministes et fournissent toujours la même solution finale, alors que d'autres sont stochastiques : deux exécutions donnent généralement des résultats différents.

Les méthodes de résolution peuvent être classées selon leur principe de fonctionnement :

- méthodes constructives ;
- méthodes basées sur une décomposition ;
- méthodes énumératives ;
- méthodes amélioratrices.

Il existe un grand nombre de méthodes dans chacune de ces catégories, aussi nous exposons simplement les grands principes de chaque catégorie :

Comme leur nom le suggère, les méthodes *constructives* complètent, à chaque itération, une solution partielle pour aboutir à une solution complète.

Par rapport aux méthodes constructives, les méthodes *basées sur une décomposition*, fonctionnent selon un principe inverse : le problème à résoudre est décomposé en sous-problèmes, supposés plus faciles à résoudre. Il existe de nombreuses méthodes basées sur ce principe, elles diffèrent essentiellement sur le type de découpage effectué.

Les méthodes *énumératives* parcourent l'espace de recherche en énumérant partiellement (ou totalement) les points de l'espace de recherche. Contrairement aux méthodes constructives qui manipulent des solutions partielles, les méthodes énumératives travaillent sur des solutions complètes.

Les méthodes *amélioratrices* consistent à générer une solution initiale s_0 et à se déplacer à chaque itération d'une solution s_i vers une solution s_{i+1} appartenant au voisinage de s_i. Nous considérons uniquement les méthodes amélioratrices pour lesquelles s_0, s_i, et s_{i+1} sont des solutions réalisables. Dans la suite, nous désignons la fonction qui calcule s_{i+1} à partir de s_i, sous le terme *opérateur* :

[1] Une solution acceptable est une solution optimale ou une solution réalisable dont le coût est suffisamment proche de l'optimum pour être retenue comme solution au problème (dans le cas où une solution optimale n'a pu être engendrée).

Définition 4 (opérateur)
Soit $\mathcal{E}$ l'ensemble des solutions réalisables. Un opérateur $\mathcal{O}$ est une fonction de $\mathcal{E}$ dans lui-même[2] *permettant de passer d'une solution s_i à une solution s_{i+1} dans l'espace de recherche.*

De nombreuses méthodes de résolution sont basées sur des méthodes amélioratrices. Elles se distinguent essentiellement par le choix de la solution s_{i+1} : ce choix peut être aléatoire, basé sur un calcul, ou réalisé après une énumération complète des solutions s_{i+1} accessibles depuis s_i.

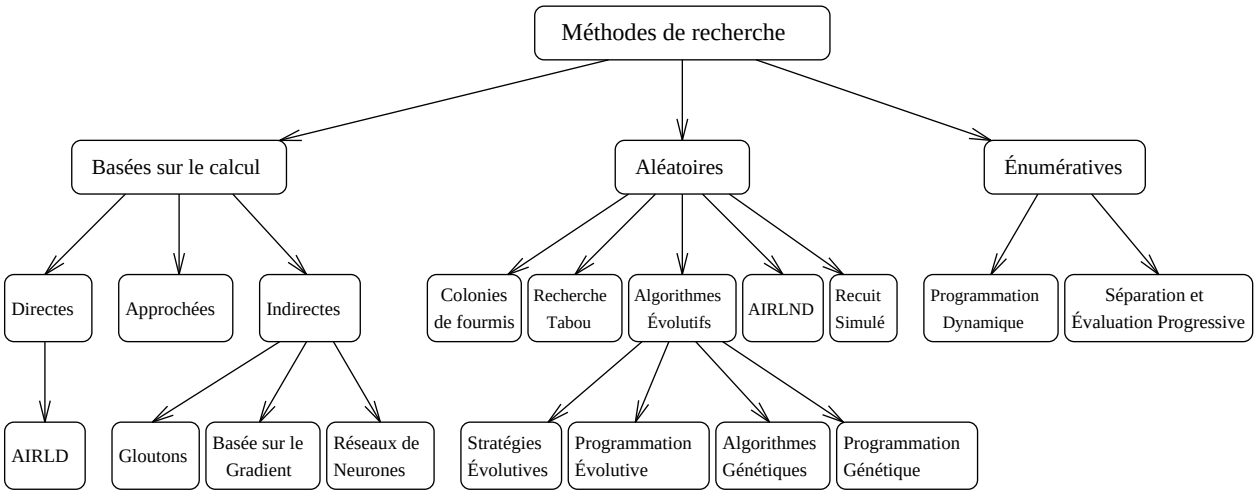


Figure 2.2: Classification des méthodes de recherche (inspirée de [Ste92, p.66]).

Nous utilisons la classification suivante pour introduire les méthodes de résolution étudiées :

- méthodes basées sur le calcul ;
- méthodes énumératives ;
- méthodes aléatoires.

Les méthodes de recherche *basées sur le calcul* utilisent des informations sur le problème traité au travers des propriétés des solutions optimales. Elles sont divisées en trois catégories. Les *méthodes approchées* utilisent généralement une méthode de relaxation pour obtenir une approximation de la valeur du coût optimum, ou une solution dont le coût est proche de la valeur optimale. Les *méthodes indirectes* utilisent des informations telles que le gradient de la fonction à optimiser pour résoudre le problème. Les *méthodes directes* utilisent (uniquement) la valeur du coût pour se guider dans l'espace de recherche.

Les méthodes *énumératives* ont déjà été décrites précédemment ; ces méthodes requièrent des ressources informatiques très importantes dès que la taille du problème traité augmente, allant même jusqu'à dépasser les capacités actuelles des calculateurs.

Les méthodes de recherche *aléatoires* parcourent l'espace de recherche de manière non exhaustive, guidées par la fonction coût ainsi que par des informations complémentaires. Ces informations secondaires sont utilisées pour « sortir » des optima locaux, ou éviter les cycles courts.

Loin d'être exhaustive, cette classification est néanmoins suffisante pour situer les unes par rapport aux autres les méthodes de recherche abordées dans les prochains chapitres (voir figure 2.2).

Dans la suite de ce document, nous nous intéressons principalement aux méthodes amélioratrices.

[2]dans cette section, nous nous restreignons à l'étude des opérateurs d'arité 1, mais il est évidemment possible de définir des opérateurs d'arité quelconque.

2.3 Critères de comparaison

L'utilisation de plusieurs méthodes pour résoudre un même problème nous amène naturellement à la comparaison de ces méthodes. Les principaux critères de comparaison sont :

- la performance (moyenne, minimale, maximale), en terme de coût de la solution finale ;
- la complexité des algorithmes ;
- l'utilisation des ressources informatiques (mémoire, temps) ;
- le nombre d'appels à la fonction coût (nombre d'évaluations) ;
- la capacité d'exploration et d'exploitation de l'espace de recherche.

La capacité d'exploration d'une méthode de recherche caractérise sa faculté à se déplacer dans l'espace de recherche sans se concentrer sur une zone particulière et en ne négligeant aucune région de cet espace.

La capacité d'exploitation d'une méthode de recherche caractérise sa capacité à se focaliser sur une zone donnée de l'espace de recherche afin de découvrir une solution dans cette zone. Par abus de langage on parle de la capacité d'une méthode à exploiter l'espace de recherche, en fait il s'agit de sa capacité à exploiter une zone de cet espace.

Certains algorithmes approchés garantissent l'obtention d'une solution de coût inférieur à une borne prédéterminée (en général exprimée en pourcentage d'excès par rapport à la valeur optimale). Mais il est assez rare que ces bornes soient satisfaisantes en pratique car ces algorithmes sont bien souvent coûteux (en temps et en ressources informatiques) et les bornes sont généralement éloignées des valeurs optimales. D'autre part, l'utilisation de méthodes heuristiques ne permet pas, en général, de calculer une telle borne alors qu'en pratique ces méthodes s'avèrent très efficaces (en terme de coût de solution finale). De ce fait, l'analyse des performances d'une méthode de recherche commence par une étude statistique des coûts des solutions finales, afin de déterminer non seulement la meilleure solution trouvée, mais encore la reproductibilité d'un tel résultat, les performances moyennes, et plus généralement la répartition des coûts des solutions finales.

Un autre critère de comparaison est la durée d'exécution des algorithmes mis en œuvre. Cependant l'utilisation de ce critère est assez délicate car elle conduit bien souvent à des comparaisons erronées. C'est le cas lorsqu'une méthode x optimisée au maximum est comparée à une implantation standard (non optimisée) d'une méthode y [BDF97]. Ou encore lorsque deux implantations d'une même méthode ont été comparées sur des calculateurs et des environnements totalement différents en utilisant un ratio censé être représentatif de la différence de puissance des calculateurs utilisés.

Le nombre d'évaluations est un critère de comparaison très intéressant car il est indépendant du calculateur utilisé. Il permet également d'évaluer l'évolution du nombre d'évaluations en fonction de la taille du problème traité pour différentes méthodes de recherche.

L'utilisation de plusieurs méthodes de recherche sur différents problèmes met en évidence les propriétés intrinsèques des méthodes mises en œuvre : certaines sont très efficaces pour explorer l'espace de recherche, alors que d'autres sont plus efficaces pour exploiter l'espace de recherche. Bien qu'il existe des ouvrages, consacrés aux différentes méthodes de recherche, extrêmement précieux lorsqu'il s'agit d'appliquer efficacement une méthode à un problème particulier, le comportement des méthodes de résolution est en grande partie mal connu : si nous savons par expérience que telle ou telle méthode est très efficace lorsqu'il s'agit d'explorer ou d'exploiter l'espace de recherche, nous sommes en revanche bien souvent incapables de prédire ou d'expliquer le comportement de cette méthode. De ce fait, comparer ces méthodes entre elles n'est pas simple.

Pour mener à bien notre étude, nous appliquons différentes méthodes de recherche à un problème particulier afin de tenter de dégager des caractéristiques qui vont nous permettre de comprendre, d'expliquer, et de projeter sur d'autres problèmes le comportement de ces méthodes. L'objectif est de déterminer les circonstances dans lesquelles une méthode donnée s'avère efficace. Ces circonstances sont d'une part dictées par l'utilisation qui est faite de la méthode de recherche (utilisée seule ou dans une méthode hybride). Elles sont d'autre part dictées par le problème au travers de la « structure » de l'espace de recherche associée. La notion de *paysage adaptatif*, exposée ci-dessous permet d'obtenir des informations précieuses sur cette *structure*.

2.3.1 Paysage adaptatif

Le paradigme de *paysage adaptatif* est un outil très utile lorsqu'il s'agit de comparer des méthodes de recherche entre elles car il permet de définir de manière formelle le comportement de ces méthodes [PRF97a, FRP97a, FPRT98, FPT97, Bru97, MB96b, MdWS91, JF95a, Sch90, REA95, DW94, Hor94, HG94, FTP97, FRPT99, Jon95].

Définition 5 (paysage adaptatif)
Soient un espace de recherche $\mathcal{E}$, un opérateur $\mathcal{O}$, et une fonction coût $f : \mathcal{E} \longmapsto \mathbb{R}$, un paysage $\mathcal{P}$ est défini par le triplet $\langle \mathcal{E}, \mathcal{O}, f \rangle$. Les points sont reliés entre eux par un graphe $G = \langle \mathcal{E}, \mathcal{A} \rangle$ où les nœuds sont les points de l'espace de recherche. L'ensemble des arcs $\mathcal{A}$ est constitué de la manière suivante : soient deux nœuds du graphe $s, s' \in \mathcal{E}$, il existe un arc de s vers s' si $s' = \mathcal{O}(s)$. Le paysage adaptatif $\mathcal{P}$ est obtenu en associant à chaque point $s \in \mathcal{E}$ une altitude correspondant à son coût $f(s)$.

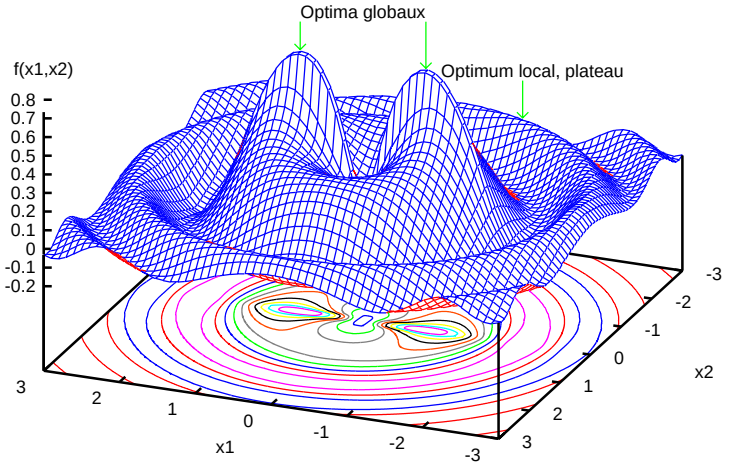


Figure 2.3: Exemple de paysage adaptatif contenant deux optima globaux, et un ensemble d'optima locaux composant des plateaux organisés de manière quasi-concentrique autour des optima globaux.

Comme le montre la figure 2.3, cette notion permet de se représenter l'espace de recherche parcouru par la méthode de recherche – pour un opérateur donné – comme un paysage composé de plateaux, de vallées, de pics, ...

Cheminer dans l'espace de recherche pour trouver une solution amène tout naturellement à la définition d'une notion de distance parcourue dans cet espace :

Définition 6 (distance-opérateur)
Pour un opérateur $\mathcal{O}$ donné, la distance-opérateur d entre deux solutions s et s' est le nombre minimum d'applications de l'opérateur $\mathcal{O}$ nécessaires pour transformer s en s'.

Par abus de langage, nous utiliserons volontiers le terme *distance*[3] pour désigner en réalité la distance-opérateur.

Selon l'objectif à atteindre (maximiser ou à minimiser le coût), un optimum local se présentera respectivement comme un pic ou un creux dans le paysage.

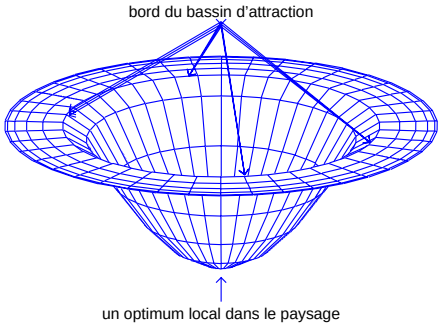


Figure 2.4: Optimum local entouré de son bassin d'attraction dans un espace de recherche à trois dimensions. Si la méthode utilisée cherche à minimiser le coût – représenté par l'axe vertical – elle sera attirée vers l'optimum local dès qu'elle aura franchi le bord du bassin d'attraction sauf si un mécanisme particulier l'en empêche. La méthode de recherche se comporte comme une bille lâchée sur le bord du bassin d'attraction : dès qu'elle se rapproche du bassin, celle-ci est attirée vers le fond du creux (*i.e.* l'optimum local).

Le bassin d'attraction d'un optimum local peut être défini de la manière suivante :

Définition 7 (bassin d'attraction)
*Pour un problème de minimisation du coût C (*resp. *de maximisation du coût C), le bassin d'attraction d'un optimum local s est l'ensemble des solution s' telles qu'il existe un chemin de s' à s où les valeurs du coût associées aux solutions sont décroissantes (*resp. croissantes*). Si s'' est une solution appartenant au chemin de s' à s, $C(s'')$ croît en fonction de la distance-opérateur entre s'' et s (lorsque l'objectif consiste à minimiser C).*

La figure 2.4 donne une représentation d'un bassin d'attraction associé à un problème de minimisation du coût : le coût décroît à mesure que l'on s'approche de l'optimum local (l'axe vertical représente le coût) dans la région constituant le bassin d'attraction.

Selon le problème traité et l'opérateur utilisé, le paysage présente différents aspects (en terme de rugosité, de présence de plateaux ou de massifs centraux ...). Cet aspect a une grande influence sur la stratégie de résolution à mettre en œuvre. Une solution envisageable pour modifier l'aspect du paysage consiste à changer d'opérateur afin de se ramener à un paysage propice à la découverte des solutions optimales. L'ajout de critères secondaires dans la fonction coût peut également modifier le paysage de manière bénéfique pour une méthode de recherche.

Pour une fonction coût et un opérateur donnés, une fois que le « type » du paysage a été identifié, ce dernier peut être utilisé pour évaluer les capacités d'exploration et d'exploitation de différentes méthodes de recherche. Le paysage sert alors d'outil pour comparer les méthodes de recherche entre elles. Il permet plus particulièrement d'étudier les stratégies de sélection utilisées dans les différentes méthodes de recherche amélioratrices pour sélectionner le ou les voisins qui seront explorés lors de la prochaine itération. Nous verrons dans les prochains chapitres comment effectuer un certain nombre de mesures sur les paysages afin de comparer entre eux opérateurs ou méthodes de recherche.

[3] bien qu'il puisse exister des couples de solutions (s, s') non reliés entre-eux.

2.3.2 Théorème NFL

De manière informelle, le théorème NFL[4] [WM95, WM97] peut s'énoncer comme suit :

Théorème 1 (théorème NFL)
Pour toute paire d'algorithmes de recherche (A_1,A_2), il y a autant de problèmes pour lesquels A_1 donne de meilleurs résultats que A_2, qu'il n'y a de problèmes pour lesquels A_2 donne de meilleurs résultats que A_1.

Par conséquent, si nous n'intégrons pas d'heuristique spécifique[5] au problème traité dans un algorithme de résolution, alors cet algorithme a autant de chances de donner de meilleurs résultats qu'une recherche aléatoire que d'en donner de moins bons (d'un point de vue théorique).

En terme de comparaison de méthodes de recherche, ce théorème permet d'affirmer qu'il n'existe pas de méthode de recherche *meilleure* que les autres pour toutes les instances de l'ensemble des problèmes.

2.4 Conclusion

Les méthodes de résolution sont extrêmement nombreuses. Elles sont basées sur des principes totalement différents, chacune explore et exploite l'espace de recherche selon des techniques qui lui sont propres. Pour un problème donné et une méthode donnée, les résultats obtenus peuvent être très hétérogènes en terme de coût selon les instances traitées : pour un ensemble d'instances données, une heuristique extrêmement simple peut s'avérer beaucoup plus efficace qu'une méthode de recherche complexe, alors qu'elle donne des résultats médiocres sur d'autre instances. Comparer ces méthodes entre elles n'est pas chose facile. Le théorème NFL permet toutefois d'affirmer qu'il n'existe pas de méthode de recherche qui soit véritablement plus performante qu'une autre sur l'ensemble des problèmes. D'un point de vue pratique, pourvu qu'elle intègre des méthodes spécifiques au problème traité, aucune méthode de recherche ne semble surpasser les autres sur toutes les instances du problème. Aussi dans les chapitres suivants nous nous concentrons sur l'étude de deux méta-heuristiques largement utilisées : la recherche tabou et les algorithmes évolutifs.

Nous avons choisi d'étudier des méta-heuristiques et non des méthodes dédiées à tel ou tel problème afin que nos travaux puissent être appliqués à d'autres problèmes sans remettre en cause la démarche suivie.

D'une part, nous avons retenu la recherche tabou pour notre étude car cette méta-heuristique est extrêmement performante dans de nombreux domaines. C'est une méthode très efficace lorsqu'il s'agit d'exploiter une zone de l'espace de recherche. D'autre part, nous avons choisi d'étudier les algorithmes évolutifs car il sont très efficaces lorsqu'il s'agit d'explorer l'espace de recherche.

Les caractéristiques intrinsèques de ces deux méthodes nous amènent naturellement à les utiliser conjointement au sein de différentes méthodes hybrides de manière à obtenir le meilleur compromis possible entre exploration et exploitation de l'espace de recherche. Nous reviendrons en détail sur ces méthodes dans les chapitres suivants. Nous allons tout d'abord effectuer un tour d'horizon des principales méta-heuristiques.

[4] NFL est une abréviation de *No Free Lunch*.
[5] ou plus généralement de méthode dédiée au problème.

Chapitre 3

Méta-heuristiques

Les méta-heuristiques sont des algorithmes qui permettent de rechercher dans un espace $\mathcal{E}$ un point remarquable p. Ce dernier est un optimum d'une fonction f matérialisant l'instance du problème Λ à résoudre. Contrairement aux algorithmes dédiés à des problèmes spécifiques, ces algorithmes sont généraux et permettent de résoudre des problèmes très divers. Ces méthodes heuristiques ne garantissent pas l'obtention d'une solution optimale ; bien qu'en pratique, elles permettent généralement de trouver une solution acceptable au problème posé en un temps raisonnable.

Il existe un grand nombre de méta-heuristiques dont certaines sont inspirées de processus naturels (recuit simulé, algorithmes génétiques, colonies de fourmis, ...) [Tai93b, OK95, Ree95, OL96, RSORS96, Lév94]. Dans notre présentation, nous nous limitons aux plus connues en insistant sur les méthodes amélioratrices.

Avant d'utiliser une méta-heuristique pour résoudre un problème donné, il est nécessaire de passer par une phase de modélisation (voir figure 2.1, page 11) qui consiste principalement à définir un *codage* et des *opérateurs* associés.

Dans ce chapitre, nous étudions tout d'abord les propriétés des codages et des opérateurs, puis nous passons en revue les principales méta-heuristiques.

3.1 Les codages

L'une des étapes de la phase de modélisation (voir figure 2.1), consiste à définir un « codage » des solutions potentielles sous une forme qui permette à la méthode de résolution de les optimiser au cours du temps :

Définition 8 (codage)
Considérons l'ensemble des solutions au problème traité et l'ensemble des configurations représentables par la structure de données manipulée par la méthode de résolution. Un « codage » est l'établissement d'une fonction injective[1] *de l'ensemble des solutions à l'ensemble des configurations.*

La figure 3.1 est un exemple de codage, donné à titre indicatif, d'algorithmes chargés de construire une solution initiale pour un problème d'ordonnancement de type *jobshop* (JSP).

Les codages peuvent être répartis en deux catégories principales : les codages indirects et les codages directs.

Un codage est indirect si la structure de données manipulée par la méthode de recherche ne représente pas directement une solution du problème traité : soit cette structure ne contient pas

[1] Deux solutions différentes correspondent nécessairement à deux configurations différentes.

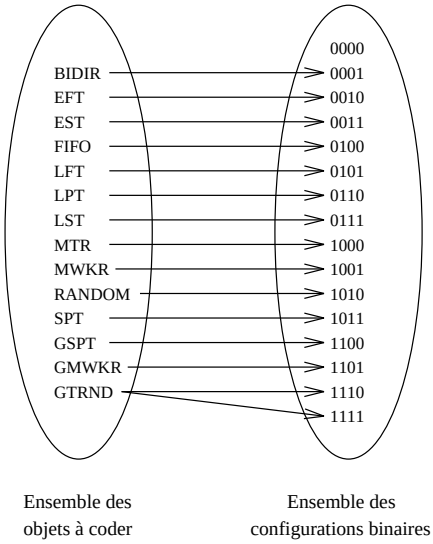
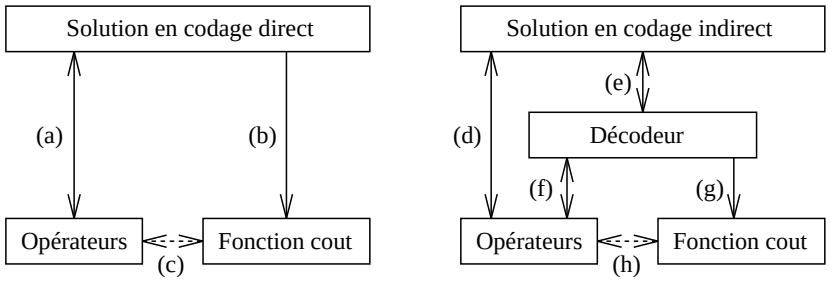


Figure 3.1: Dans cet exemple de codage, l'ensemble des objets à coder est constitué d'algorithmes utilisés pour construire une solution initiale pour un problème d'ordonnancement de type *jobshop* (voir 6.1.4, page 72). Ces algorithmes sont codés sous forme de configurations binaires.

toutes les informations nécessaires à la reconstitution d'une solution réalisable ; soit les données sont exprimées sous une forme non directement exploitable pour évaluer la solution.



En codage direct, opérateurs (a) et fonction coût (b) utilisent directement les informations contenues dans la structure de données représentant la solution potentielle. Les opérateurs utilisent éventuellement des informations issues du calcul de la fonction coût (c) pour agir sur la structure de données.

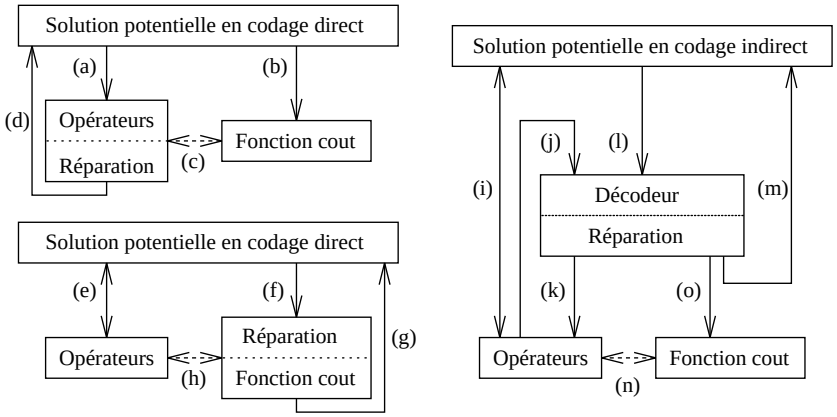
En codage indirect, opérateurs (f) et fonction coût (g) utilisent les informations issues du décodage (e) de la structure de données. Les opérateurs peuvent aussi agir au niveau de la structure de données (d) sans passer par le décodeur.

Figure 3.2: Codage direct, codage indirect, opérateur et fonction coût.

Dans le cas d'un codage indirect, il est nécessaire de décoder la structure de données, c'est-à-dire de la transformer en une solution réalisable, en ajoutant ou en modifiant éventuellement des informations : c'est le rôle du *décodeur*. La figure 3.2 illustre les différentes stratégies utilisables : les opérateurs agissent soit sur la structure décodée (f) soit sur la solution codée (d), en utilisant éventuellement la valeur du coût (h). Ce coût est évalué à partir de la solution décodée (g).

Dans un codage direct, la structure de données manipulée par la méthode de recherche représente exactement une solution au problème. Il n'est pas nécessaire d'utiliser un décodeur. La méthode de recherche dispose de toutes les informations sur les solutions, puisqu'elles sont directement intégrées dans la structure de données manipulée. De ce fait, il est possible de définir des opérateurs dépendants du domaine, capables de réduire le temps nécessaire à la découverte de bonnes solutions

en tirant profit de l'ensemble des informations intégrées dans la structure de données [LRS+98b]. Comme le montre la figure 3.2, les opérateurs agissent « directement » sur la solution potentielle (a) en utilisant éventuellement la valeur du coût (c). Le coût est évalué à partir de la solution potentielle sans nécessiter de décodage (b).



En codage direct, la réparation peut prendre place dans les opérateurs ou dans la fonction coût.
Dans le premier cas, les opérateurs (a) engendrent une solution réalisable après réparation (d). La fonction coût (b) utilise directement les informations contenues dans la structure de données représentant la solution potentielle.
Dans le second cas, la réparation est incluse dans la phase d'évaluation. Les opérateurs (e) agissent directement sur la structure de données.
Les opérateurs utilisent éventuellement des informations relatives au coût (c,h).

En codage indirect, le cas le plus fréquent consiste à intégrer le mécanisme de réparation au sein du décodeur. Les opérateurs peuvent agir sur des solutions « non réparées » (i), ou au contraire sur des solutions décodées et réparées (j,k). L'évaluation d'une solution (l,o) ne peut se faire qu'après décodage et réparation. Il est possible de réinjecter le résultat de la phase de réparation dans la structure de données (m). Les opérateurs utilisent éventuellement des informations issues du calcul de la fonction coût (n) pour agir sur la structure de données.

Figure 3.3: Intégration d'un mécanisme de réparation.

Nous considérons à nouveau deux types de codages distincts : les codages n'autorisant que des solutions réalisables et les codages autorisant des solutions invalides.

Le premier cas (où le codage assure que les solutions sont réalisables) correspond à la figure 3.2, déjà étudiée.

Dans le second cas, les opérateurs ou le décodeur sont chargés de maintenir la validité des solutions engendrées. La figure 3.3 illustre les différentes stratégies utilisables selon le type (direct ou indirect) de codage. En cas de codage direct, deux possibilités sont envisageables : soit les opérateurs intègrent un mécanisme de réparation qui maintient la validité des solutions, soit la phase d'évaluation intègre un mécanisme de réparation appliqué avant le calcul du coût. En cas de codage indirect, il est nécessaire de décoder la solution avant d'évaluer son coût si bien que le mécanisme de réparation est généralement intégré au décodeur. Selon la stratégie retenue, les opérateurs manipulent les solutions (i) non décodées, ou après décodage et réparation (j,k) en utilisant éventuellement des informations liées au coût de la solution (n). Il existe des variantes qui permettent de simplifier tel ou tel composant : par exemple le fait de réinjecter systématiquement dans la structure de données (m) le résultat de la phase de réparation peut simplifier les opérateurs.

Une fois les principales catégories de codages définies, nous présentons les propriétés associées aux codages. La première est liée à la redondance des configurations, elle permet de caractériser

deux familles de codages : les codages redondants et les codages non redondants.

Dans le premier cas, plusieurs valuations de la structure de données représentent la même solution. Bien souvent ce type de situation est un effet de bord indésirable d'un décodeur, mais cette situation est parfois recherchée pour favoriser certains types de solutions représentées par différentes valuations au détriment des autres.

Dans la majorité des cas, l'utilisation d'un codage non redondant est plus appropriée : les redondances ne font qu'augmenter artificiellement la taille de l'espace de recherche du problème traité [ERR94].

Enfin, une propriété importante caractérise l'interdépendance des éléments de base constituant un codage : l'épistasie [FRP98, RVSK97, BBM93c, Roc98]. Cette propriété a une grande influence sur le choix des stratégies à adopter pour résoudre le problème posé :

- dans un codage non épistatique, il est possible de faire varier indépendamment les éléments de base constituants la structure de données afin d'atteindre la valeur de coût optimale ;
- dans un codage fortement épistatique, la structure de données doit être considérée dans son ensemble car la variation d'un élément de base influe sur la valeur des autres éléments.

Cette propriété est indirectement liée au problème traité : en effet selon le problème, il sera plus ou moins facile de définir un codage faiblement épistatique. Par exemple les paysages NK ont été définis de manière à pouvoir faire varier le degré d'épistasie. L'utilisation de différents couples ⟨codages, opérateurs⟩ permet également de modifier l'épistasie [BBM93c].

3.1.1 Discussion

Un « bon » codage assure un compromis entre la quantité d'information intégrée dans le codage et la complexité des opérateurs : un codage contenant beaucoup d'informations tend vers un codage direct, ce qui – en général – simplifie le décodeur, mais complique les opérateurs. En revanche, un codage contenant peu d'informations tend vers un codage indirect, ce qui complique le décodeur mais simplifie les opérateurs. Quel que soit le codage indirect utilisé, le rôle du décodeur est très important. Cependant les études sur le sujet sont assez rares [HKB94, HR98] alors qu'il peut à lui seul interdire la découverte d'une solution optimale si certaines solutions ne sont pas accessibles à l'issue du décodage [PG95].

Il est également nécessaire de trouver un compromis entre complexité des opérateurs et codage de solutions invalides ou redondantes. En effet, la présence de solutions invalides influence les autres composants de la méthode de recherche. Ainsi, l'évaluation des solutions invalides peut provoquer des effets de bord indésirables : par exemple, si la sélection est de type élitiste, la meilleure solution rencontrée au cours de la recherche peut être une solution invalide [Mic96]. De plus, il faut que les opérateurs puissent être appliqués à des solutions invalides. Enfin, la taille de l'espace de recherche se trouve augmentée, puisqu'il est alors constitué de l'ensemble des solutions réalisables et de l'ensemble des solutions invalides.

Le tableau 3.1 synthétise les avantages et inconvénients généralement obtenus en fonction des différentes caractéristiques des codages.

Les redondances sont bien souvent liées au décodeur employé pour transformer une structure de donnée (codage indirect) en une solution : la « fonction de décodage » n'est pas injective (ce qui va à l'encontre de la définition d'un codage qui est *normalement* une fonction injective). En pratique, un codage redondant n'est pas nécessairement synonyme de détérioration de performances. Tout dépend en effet du taux de solutions redondantes, du problème traité, des opérateurs utilisés ... Par

	direct	indirect
présence de solutions invalides (réparation)	absence de décodeur ; opérateurs plus complexes qu'en codage indirect ; les solutions invalides augmentent la taille de l'espace de recherche sauf si les solutions potentielles sont systématiquement réinjectées après réparation ; le mécanisme de réparation complique les opérateurs ou l'évaluation	nécessite un décodeur ; les solutions invalides augmentent la taille de l'espace de recherche sauf si les solutions potentielles sont systématiquement réinjectées après réparation ; le mécanisme de réparation complique les opérateurs ou l'évaluation
absence de solutions invalides (sans réparation)	absence de décodeur ; opérateurs plus complexes qu'en codage indirect ; l'absence de solutions invalides est susceptible de simplifier les opérateurs	nécessite un décodeur ; l'absence de solutions invalides est susceptible de simplifier les opérateurs

redondant	non redondant
augmente la taille de l'espace de recherche ; susceptible de simplifier le décodeur	la taille de l'espace de recherche correspond au nombre de valuations possibles de la structure de donnée

faible épistasie	forte épistasie
l'indépendance des éléments de base est susceptible de simplifier la méthode de recherche	selon le problème, un codage fortement épistatique peut être plus facile à découvrir ou à mettre en œuvre

Tableau 3.1: Synthèse des caractéristiques des codages dans le cas général.

contre, si l'on n'y prend garde, il peut arriver que la fonction de décodage ne soit pas surjective : c'est-à-dire que certaines solutions réalisables ne soient jamais atteintes. Lors de la définition d'un décodeur, il faut s'assurer que cette situation ne puisse pas se produire, faute de quoi la méthode de recherche intégrant ce décodeur risque de chercher indéfiniment une solution optimale.

Dans le paragraphe précédent, le problème de redondance est évoqué dans le cas d'un codage indirect, car ce problème survient essentiellement avec ce type de codage : en principe, à une valuation de la structure de données correspond exactement une solution en codage direct. Cependant le problème de redondance peut survenir si le problème traité comporte, par exemple, des symétries. Donc, de manière générale, un codage direct peut tout à fait être redondant.

Enfin, un codage fortement épistatique complique la tâche de la méthode de recherche : cette dernière doit considérer chaque solution potentielle dans son ensemble pour tenir compte des interactions entre les éléments de base constituants la solution.

Une fois le codage défini, il est nécessaire de considérer les opérateurs associés[2].

3.2 Les opérateurs

Une fois le codage établi, il faut définir le ou les opérateurs qui vont permettre de passer d'une solution à une autre, ou de construire une solution. Ces opérateurs doivent garantir un certain nombre de conditions pour fournir une méthode de recherche performante. Dans cette section nous étudions les propriétés des opérateurs après avoir introduit la terminologie associée à cette étude.

Dans les méthodes amélioratrices, la notion de voisinage est particulièrement importante car, du déplacement de la méthode de recherche dans le voisinage vont dépendre en grande partie les performances obtenues. La notion de « voisin » est définie par l'opérateur :

[2]Ces deux étapes (définition d'un codage, définition des opérateurs) sont bien souvent regroupées en une seule étant donné les très fortes interactions entre codage et opérateurs.

Définition 9 (voisin)
Un voisin s', ou solution voisine, est une solution obtenue par une (et une seule) application de l'opérateur de déplacement à la solution courante s.

À partir de la définition précédente, nous définissons le *voisinage* d'une solution :

Définition 10 (voisinage)
Le voisinage de la solution courante s est l'ensemble des voisins s' obtenus par une application de l'opérateur $\mathcal{O}$ à s. Il est noté $V_{\mathcal{O}}(s)$.

Dans le cas où l'opérateur engendre systématiquement la même solution voisine s' à partir de s, le voisinage est restreint au point s' : $V_{\mathcal{O}}(s) = \{s'\}$. Par contre, si l'opérateur est basé sur un mécanisme probabiliste ou sur un paramétrage influant sur le point engendré, alors le voisinage est constitué de plusieurs voisins (*i.e.* $|V_{\mathcal{O}}(s)| \geq 1$). Par exemple, si nous considérons des opérateurs dont le principe de fonctionnement consiste à permuter deux composantes d'un vecteur s, plusieurs solutions voisines s' seront engendrées si les composantes à permuter sont choisies de manière aléatoire.

Dans la suite de mémoire, nous supposons que les opérateurs utilisés sont susceptibles d'engendrer plusieurs solutions voisines distinctes.

La taille du voisinage $|V_{\mathcal{O}}(s)|$ engendré par un opérateur $\mathcal{O}$ définit le degré de connexité entre les solutions, c'est-à-dire le nombre de solutions situées à distance « un » en terme de nombre d'applications de l'opérateur $\mathcal{O}$. Plus cette taille est importante, plus les solutions sont « proches » les unes des autres.

La méthode de recherche dispose au minimum d'une « vue » sur le voisinage de la solution courante au travers du coût des voisins. Cette vue très limitée de l'espace de recherche, bien souvent complétée par des mécanismes de mémorisation à court et à long terme, va guider la méthode de recherche au cours de son cheminement dans l'espace de recherche. Ce *cheminement* est obtenu par applications successives de l'opérateur à une solution[3] :

Définition 11 (chemin)
Un chemin de longueur n d'origine s_1 est une suite de solutions $(s_1, s_2, \ldots, s_n)$ telle que $\forall i\, 1 \leq i < n$, s_{i+1} est obtenue par application de l'opérateur à s_i.

Cheminer dans l'espace de recherche ne suffit pas à garantir la découverte d'un optimum global : l'opérateur doit non seulement être capable d'engendrer une solution optimale, mais également permettre d'accéder à une solution optimale depuis n'importe quelle solution initiale. Cette propriété est appelée la condition d'accessibilité d'un optimum global [TCM95, HW96] :

Définition 12 (condition d'accessibilité)
Un opérateur garantit la condition d'accessibilité d'un optimum global s'il existe au moins un chemin depuis n'importe quelle solution initiale admissible s_o vers une solution optimale s^*.

Il est souhaitable de s'assurer que l'opérateur garantisse la condition d'accessibilité d'un optimum global dans une méthode de recherche utilisant un seul opérateur pour explorer l'espace de recherche. Cette notion d'accessibilité est à rapprocher de la notion d'ergodicité :

Définition 13 (opérateur ergodique)
Un opérateur est ergodique s'il permet d'atteindre n'importe quel point de l'espace de recherche $\mathcal{E}$ à partir d'une solution initiale admissible s_o en appliquant cet opérateur un certain nombre de fois à s_o.

[3] ou plusieurs

Remarque : Un opérateur ergodique vérifie la condition d'accessibilité (mais la réciproque n'est pas nécessairement vraie).

Une fois définies les principales propriétés associées aux opérateurs, nous présentons les principales catégories d'opérateurs. Dans un premier temps, nous pouvons distinguer deux types d'opérateurs :

- opérateurs n'engendrant que des solutions réalisables ;
- opérateurs engendrant des solutions réalisables ou invalides.

Dans le second cas, il est souhaitable d'utiliser un mécanisme de réparation afin d'éviter à la méthode de recherche de cheminer trop longtemps dans l'espace des solutions invalides. L'utilisation systématique d'un mécanisme de réparation à l'issue de l'application de l'opérateur (voir figure 3.3, page 21) permet de se ramener au premier cas où seules existent des solutions réalisables, ce qui a pour avantage de simplifier la fonction coût puisque seules des solutions réalisables doivent être évaluées.

L'opérateur doit assurer le passage d'une solution s à une solution voisine s' en préservant (ou en améliorant) dans s' les « bonnes propriétés » de s. Autrement dit s' doit « hériter » des bonnes propriétés de s [BMK96]. Cet héritage est lié aux caractéristiques des éléments constitutifs de la structure de données utilisée pour réaliser le codage. À l'instar de [LK92], nous distinguons les caractéristiques suivantes[4] :

- la valeur : la valeur d'un ou plusieurs éléments constitutifs de s est préservée dans s' sans tenir compte de leur position. Par exemple le passage de $s = (8\ \underline{7}\ 4\ 2\ \underline{9}\ 3)$ à $s' = (\underline{7}\ 1\ 5\ 6\ 10\ \underline{9})$ préserve les éléments $\underline{7}$ et $\underline{9}$.
- la position : l'opérateur préserve autant que possible la position des éléments constitutifs. Par exemple, le passage de $s = (8\ \underline{10}\ 2\ \underline{4}\ \underline{5}\ \underline{1}\ 9)$ à $s' = (2\ \underline{10}\ 7\ \underline{4}\ \underline{5}\ \underline{1}\ 3)$ préserve la position des éléments $\underline{10}$, $\underline{4}$, $\underline{5}$ et $\underline{1}$.
- l'ordre (relatif) : dans les codages sous forme de permutations d'entiers, *l'ordre relatif* entre les éléments constitutifs du codage est une information primordiale et l'opérateur tente de le préserver. Par exemple, le passage de $s = (7\ 3\ \underline{1}\ 8\ \underline{6}\ 0)$ à $s' = (10\ \underline{1}\ 4\ 2\ 9\ \underline{6})$ préserve l'ordre des éléments $\underline{1}$ et $\underline{6}$.
- l'adjacence : l'opérateur préserve autant que possible la position des éléments constitutifs. Par exemple, le passage de $s = (1\ 4\ \underline{7}\ \underline{8}\ 6)$ à $s' = (10\ \underline{8}\ \underline{7}\ 1\ 9)$ préserve l'adjacence des éléments $\underline{7}$ et $\underline{8}$.

Certains opérateurs préservent plusieurs caractéristiques, c'est le cas notamment des opérateurs qui préservent *l'ordre absolu* [KDG92] : un opérateur préserve *l'ordre absolu* s'il préserve à la fois la position et l'ordre relatif des éléments constitutifs du codage.

Selon le problème traité, les caractéristiques héritées sont plus ou moins importantes. Ainsi [SDMW96, BMK96] montrent que pour un problème d'ordonnancement de type *jobshop*, les opérateurs basés sur un héritage de l'ordre sont mieux adaptés que des opérateurs basés sur l'héritage de l'adjacence (si l'ordre correspond au déroulement dans le temps des opérations).

[4] Dans les exemples de caractéristiques, les valeurs des nombres sont données à titre indicatif. En l'absence d'indication, il peut s'agir d'une liste de priorités, d'une permutation d'entiers traduisant l'ordre des opérations, d'une liste de numéros de règles à appliquer ...

L'héritage transmis à la solution engendrée par l'opérateur peut également s'exprimer en terme de respect des contraintes. En effet, lorsqu'il s'agit de résoudre un problème où existent un grand nombre de contraintes, il n'est pas toujours possible (ou souhaitable) de reporter l'ensemble des contraintes dans la fonction coût. Dans ce cas, les opérateurs peuvent être chargés de s'assurer du respect d'une partie des contraintes [DG94, ERR94, CDM90, BEW95, vKHHK95, RCF94a, JM91b, Mic96, MDRS96, Mic95, MA94, JW93, RAARS98, YB98]. En ce cas la solution engendrée s' doit au moins respecter autant de contraintes que la solution initiale s. En général cette condition est assurée par un mécanisme de réparation plus ou moins intégré à l'opérateur. Par exemple, l'algorithme génétique Genocop III [JM91b, vWKvdBvK94, Mic96, MDRS96] dédié à la résolution de problèmes d'optimisation avec contraintes linéaires ou non-linéaires, utilise des opérateurs dédiés ainsi qu'un mécanisme de réparation.

Pour évaluer les performances de l'opérateur, nous effectuons deux types de mesures :

- les performances de l'opérateur seul ;
- les performances de l'opérateur intégré à la méthode de recherche.

Les performances de l'opérateur seul nous permettent d'évaluer les performances *a priori* de l'opérateur avant même son utilisation dans une méthode de recherche. Nous effectuons plus particulièrement des mesures de longueur de corrélation : cette mesure consiste à évaluer la corrélation entre le coût d'une solution initiale et le coût d'une solution obtenue après un certain nombre d'applications de l'opérateur à partir de cette solution initiale. Il a été montré expérimentalement qu'il existe un lien entre la longueur de corrélation et les performances de l'opérateur.

Le second type de mesures permet d'évaluer les interactions avec les autres composants de la méthode de recherche utilisée.

Dans chaque cas (opérateur seul ou intégré dans la méthode de recherche), d'autres critères peuvent être évalués. Il s'agit du gain entre s et s' (s' est obtenue après application de l'opérateur à s). Le gain est le ratio entre le coût de s et le coût de s', il permet d'estimer la variation du coût induite par l'application de l'opérateur considéré.

3.2.1 Discussion

La taille du voisinage[5] engendré par l'opérateur joue un grand rôle : elle influe sur le nombre de choix qui s'offrent à une méthode de recherche donnée pour passer de la solution courante à une solution voisine. Plus ce nombre est élevé, plus la méthode de recherche à de chances de pouvoir s'échapper d'un optimum local en trouvant une solution voisine meilleure que la solution courante. *A contrario* il est toujours possible de définir un voisinage restreint lorsque la taille de ce voisinage devient trop grande.

Une propriété importante, nommée *héritage* est définie par ce qui est « transmis » par l'opérateur lors du passage de la solution courante à la solution suivante : est-ce une information liée à la position, ou à la valeur des éléments constitutifs de cette solution initiale ? La réponse est liée au problème traité : pour des solutions codées sous forme de permutation d'entiers, la position est l'information à transmettre ; pour un problème numérique codé sous forme de vecteurs, c'est plutôt la valeur des composantes du vecteur [BMK96, SDMW96, FM91, Mic92, Mic94, JM91a].

Une mesure permet de quantifier l'efficacité moyenne d'un opérateur : le gain (en terme de coût) entre la solution courante et la meilleure[6] solution voisine. Cette mesure de gain peut être effectuée

[5] Voir définition 10, page 24.

[6] en terme de coût.

au cours de l'évolution d'une méthode de recherche. Elle ne suffit pas à caractériser un opérateur à elle seule, car ce dernier peut très bien offrir un gain moyen élevé sans jamais conduire la méthode de recherche à une solution optimale (c'est pourquoi la condition d'accessibilité doit être vérifiée).

De manière à limiter l'apparition de plateaux, il faut éviter, autant que possible, de définir l'opérateur pour qu'il puisse se ramener à l'identité sous certaines conditions : selon la méthode de recherche utilisée ceci peut être vu comme un plateau et déclencher des mécanismes qui détériorent les performances. Par exemple, sur des chaînes de bits, il est préférable de définir un opérateur qui inverse un bit, plutôt qu'un opérateur qui remplace ce bit par un bit de valeur aléatoire.

En général, l'utilisation d'un codage direct permet à l'opérateur de disposer du maximum d'informations sur le problème traité, ce qui offre la possibilité de créer des opérateurs dédiés performants. De tels opérateurs sont assez complexes à mettre en œuvre. Aussi, dans de nombreuses applications des méthodes de recherche, des opérateurs généraux sont utilisés en association à un codage indirect. Les résultats montrent pourtant que l'intégration d'un maximum d'informations liées au problème traité donne de meilleures performances : il s'agit de trouver un compromis entre un codage et des opérateurs simples mais « aveugles » sur le problème traité et un couple ⟨codage, opérateurs⟩ dédié au problème traité.

3.3 Algorithmes itératifs de recherche locale

Sous le terme d'algorithme itératif de recherche locale, nous pouvons désigner un grand nombre de méthodes amélioratrices basées sur un même principe :

Définition 14 (AIRL)
Un AIRL est un algorithme qui, partant d'une solution initiale, crée une nouvelle solution appartenant au voisinage de la solution courante, et itère le processus tant que la solution est améliorée. Si l'algorithme ne parvient pas à améliorer la solution au bout d'un certain nombre de tentatives, l'algorithme est stoppé.

```
Créer une solution initiale s
Répéter ...
   Calculer V_O(s), le voisinage de s
   Choisir un voisin s'∈V_O(s)
   Si f(s') meilleure que f(s) alors
      s←s'
Tant que critère-d'arrêt non satisfait
Afficher("Solution trouvée " s)
```

Figure 3.4: Principe de fonctionnement d'un algorithme itératif de recherche locale.

La figure 3.4 présente un pseudo-algorithme correspondant au principe de fonctionnement d'un AIRL. De nombreuses méta-heuristiques peuvent se ramener à ce pseudo-algorithme (notamment les méta-heuristiques étudiées dans ce chapitre).

Il existe des méthodes de recherche extrêmement simples basées sur ce pseudo-algorithmes, plus communément désignées sous le terme « hill-climber » [OL96, VAL92, JPY88, Tai93b, OK95, Pre99]. Un *hill-climber* est une heuristique très simple qui requiert peu de ressources (en terme de temps et de mémoire). En revanche, elle est facilement piégée par un optimum local. Il peut sembler curieux d'étudier un algorithme aussi simple ; pourtant il s'avère parfois être aussi performant que d'autres méthodes lors de la résolution de certains problèmes [Jon94, MH93a, JW94, DPT96a, DPT95c,

DPT96b, DPF+98, DPT+98, DBB94, FGL96, Pre99]. Mais surtout, il peut servir de référence pour comparer les performances des autres méthodes de recherche plus complexes, donc supposées être plus performantes.

Le point délicat lors de l'utilisation d'un *hill-climber* est le choix de l'opérateur de déplacement car c'est uniquement de lui que vont dépendre les performances de l'algorithme. De ce fait, le *hill-climber* est un excellent outil lorsqu'il s'agit de comparer les performances de plusieurs opérateurs.

Dans le cadre de notre étude, nous utilisons deux types de *hill-climbers* : les algorithmes itératifs de recherche locale déterministes (AIRLD) et les algorithmes itératifs de recherche locale non déterministes (AIRLND).

Définition 15 (AIRLD)
Un AIRLD est un AIRL déterministe : le choix du voisin s' de la solution courante s est déterministe (voir figure 3.4, page 27).

```
critère-d'arrêt←faux
Créer une solution initiale s
Répéter ...
   Calculer V_O(s), le voisinage de s
   Choisir un voisin s' tel que f(s') = min_{s'∈V_O(s)} f(s')
   Si f(s') < f(s) alors
      s←s'
   Sinon
      critère-d'arrêt←vrai
Tant que critère-d'arrêt non satisfait
Afficher("Solution trouvée " s)
```

Figure 3.5: Principe de fonctionnement d'un AIRLD.

Dans le cadre de nos travaux, nous avons défini un AIRLD dont l'objectif consiste à minimiser la fonction coût f : $f(s')$ est meilleure que $f(s)$ si $f(s') < f(s)$. Le voisin choisi s' est tel que $f(s')$ soit minimale dans $V_{\mathcal{O}}(s)$. La figure 3.5 présente un pseudo-algorithme correspondant au principe de fonctionnement de l'AIRLD utilisé dans le cadre de notre étude.

Le travail réalisé par l'AIRLD utilisé peut être défini de la manière suivante : l'AIRLD définit dans le paysage[7] $\mathcal{P}$ un chemin de longueur n $(s_1, s_2, \ldots, s_n)$ tel que $f(s_i) > f(s_{i+1})$.

La taille du voisinage pose de gros problèmes en terme de nombre d'appels à la fonction coût à l'AIRLD, tel que nous l'avons défini. Pour pallier ce problème, un voisinage restreint peut être obtenu à partir d'un critère destiné à sélectionner les voisins à visiter. Cependant, il n'est pas toujours possible (ou souhaitable) de définir un tel critère. Une solution consiste à restreindre le voisinage en choisissant *au hasard* un certain nombre de voisins, cette méthode est à la base des algorithmes itératifs de recherche locale non déterministes (AIRLND).

Les algorithmes itératifs de recherche locale non déterministes (AIRLND) constituent le second type de *hill-climbers* utilisés :

Définition 16 (AIRLND)
Un AIRLND est un AIRL non déterministe : le choix du voisin s' de la solution courante s n'est pas déterministe (voir figure 3.4, page 27).

[7] Voir définition 5, page 16.

Nous avons défini plusieurs AIRLND dont l'objectif consiste à minimiser la fonction coût f. Le plus simple d'entre-eux peut être défini en quelques mots : $f(s')$ est meilleure que $f(s)$ si $f(s') < f(s)$. Le voisin choisi s' est un voisin, tiré au hasard dans $V_{\mathcal{O}}(s)$, tel que $f(s') < f(s)$.

Définition 17 (mouvement)
Le mouvement de s à s' correspond à la génération de s' à partir d'une seule application de l'opérateur $\mathcal{O}$ à s. La notion de mouvement traduit le fait qu'à l'issue de la génération de s' à partir de s, s' devient la solution courante.

Pour affiner nos recherches, deux variantes des AIRL ont été définies. La première consiste à intégrer une mémoire (M) des mouvements récents (dans le but d'éviter les cycles courts). La seconde consiste à intégrer un mécanisme (P) de déplacement sur les plateaux. Ces variantes seront décrites en détail lors de leur première utilisation dans ce mémoire (section 7.3.3, page 108). La figure 3.6 présente une classification des AIRL utilisés.

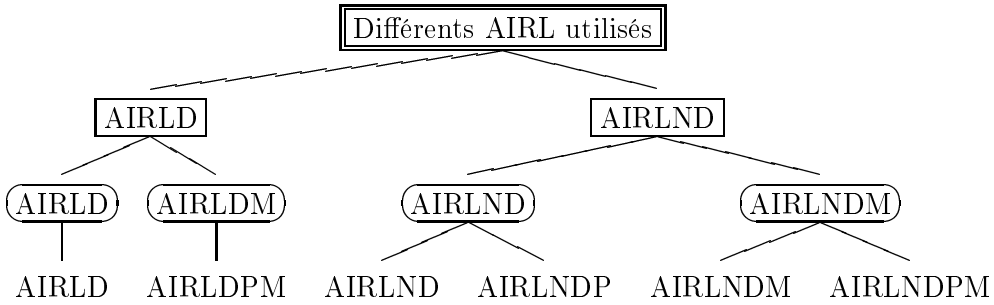


Figure 3.6: Classification des AIRL utilisés. Les AIRL utilisés sont soit déterministes (D), soit non-déterministes (ND), soit doté d'une mémoire (M) ou enfin capable de se déplacer sur les plateaux (P). En ce qui concerne les AIRLND, le voisinage n'est pas complètement calculé, par conséquent l'ajout d'une mémoire peut être utile pour stocker la fraction de $V_{\mathcal{O}}(s)$ déjà évaluée (AIRLNDM). La mémoire peut également servir pour stocker les mouvements récents (AIRLNDPM). Il est envisageable de cumuler les deux possibilités, c'est ce qui est effectivement utilisé dans ce que nous désignons sous le terme AIRLNDPM (de manière plus stricte, ces algorithmes devraient être dénommés AIRLNDMPM pour illustrer l'utilisation de la mémoire à deux niveaux). En ce qui concerne les AIRLD utilisé, soit l'AIRLD ne peut pas se déplacer sur les plateaux et dans ce cas il est inutile d'utiliser une mémoire car $V_{\mathcal{O}}(s)$ est complètement calculé (AIRLD), soit l'AIRLD est capable de se déplacer sur les plateaux et compte tenu du coût de calcul du meilleur voisin ce dernier est systématiquement doté d'une mémoire des mouvements récents pour éviter les cycles courts (AIRLDPM).

L'intégration de mécanismes complémentaires dans ces AIRL a donné naissance à tout un ensemble de méta-heuristiques performantes telles que le recuit-simulé, la recherche tabou, ou les algorithmes évolutifs.

Notation : Dans la suite de nos travaux nous sommes amenés à utiliser différentes fonctions coûts, d'où l'introduction de la notation suivante : AIRL(f) précise que l'AIRL utilise la fonction coût f.

3.3.1 Des AIRL aux méta-heuristiques

Dans la suite de ce chapitre nous étudions quelques méta-heuristiques basées sur le principe de fonctionnement des AIRL. Pour chacune d'entre elles, nous comparons leur pseudo-algorithme à celui des AIRL (figure 3.4, page 27) afin de mettre en évidence les mécanismes ajoutés à l'AIRL.

3.4 Recuit simulé

L'idée de cette méthode de recherche consiste à reproduire de manière simplifiée le phénomène de recuit obtenu lors du refroidissement d'un matériau solide préalablement chauffé : la structure du matériau obtenu (la taille des cristaux notamment) varie en fonction de la vitesse de refroidissement. Le recuit simulé reproduit les variations d'énergie observées lors d'un processus de recuit.

Contrairement à un *hill-climber*, facilement piégé par un optimum local, le recuit simulé est doté d'un mécanisme qui lui permet – sous certaines conditions – de s'échapper des optima locaux. Ce mécanisme, bien qu'élémentaire, confère au recuit simulé des caractéristiques qui donnent de très bons résultats dans de nombreux domaines d'application.

Problème	Références
ordonnancement, planification, production	[TCM95, HH97, OL96, Özd96, Bra96, MT96, Maq96, Fle95, Ihl96, Tal98, VAL96, CP96, Pen94, SAW99, CR99]
conception d'emploi du temps	[OL96, AD]
aviation, conflits ATC	[OL96]
ordonnancement d'instructions, micro-code	
constitution et ordonnancement de trains	[OL96]
construction de cartes génétiques	
routage de paquets dans un réseau	
électronique (routage, ...)	[Tal98, Rut89, GOD+98]
robotique	
économie, finance	[OL96]
chimie, biochimie	[OL96, Tal98]
reconnaissance de formes	
optimisation de réseaux (gaz, fluides, télécom.)	[OL96, KCR95]
gestion de réseau de distribution électrique	[Tal98]
optimisation de structure mécanique	
conception de matériaux composites	
conception et apprentissage de réseaux de neurones	[DN92, SH87]
jeux (taquin, ...)	
optimisation de fonction numérique	[OL96, CMMR87, Tal98, Ack87]
coloriage de graphe	[OL96]
partitionnement de graphe	[Tal98, GOD+98]
clique maximale dans un graphe	
sac à dos, problèmes de découpe	[OL96, Tal98, AFH95]
couverture d'ensemble, partitionnement d'ensemble	[OL96]
applications médicales	
problème du voyageur de commerce (TSP)	[OL96, Tal98]
problème d'affectation quadratique (QAP)	[OL96, Tal98]
problème de satisfaisabilité (SAT)	[Par95, Tal98]
introduction, état de l'art	[Dav87, Ree95, RSORS96, OL96, Tal98, Rut89, Lév94]

Tableau 3.2: Quelques domaines d'application du recuit simulé.

Rares sont les problèmes auxquels le recuit simulé n'a pas été appliqué à ce jour, aussi le tableau 3.2 ne saurait être exhaustif ...

3.4.1 Terminologie

Par analogie au processus physique dont cette méta-heuristique est inspirée, un certain nombre de termes sont définis :

Définition 18 (énergie)
L'énergie désigne la valeur de la fonction coût de la solution courante.

Définition 19 (état)
Un état correspond à une solution au problème traité.

3.4.2 Principe de fonctionnement

Le recuit simulé est un *hill-climber* auquel a été ajouté la possibilité d'accepter – sous certaines conditions – une solution plus mauvaise que la solution courante pour sortir d'un optimum local [DS87]. L'évolution de l'algorithme est gérée par un paramètre appelé *énergie* : l'énergie d'une solution est d'autant plus élevée que le coût de la solution est élevé. L'algorithme du recuit simulé est donné en figure 3.7.

```
état ← état initial
Répéter ...
  T ← nouvelle_température
  Répéter ...
    nouvel_état ← choix aléatoire d'un voisin ∈ V_O(état)
    ΔE ← E(nouvel_état) - E(état)
    Si ΔE < 0 alors
      état ← nouvel_état
    Sinon
      état ← nouvel_état avec une probabilité e^(-ΔE/kT)
    Décrémenter(T)
  Tant que T ≥ Tseuil
Tant que critère-d'arrêt non satisfait
Afficher("Solution trouvée " état)
```

Figure 3.7: Principe de fonctionnement du recuit simulé.
E(état) correspond à l'énergie de l'état considéré ; k est un facteur d'échelle, en général $k = 1$.

La température T est un paramètre de l'algorithme : elle détermine la probabilité avec laquelle une dégradation de la solution courante est acceptée. La température décroît graduellement au cours du temps. Au début de la recherche, la température élevée permet au recuit d'explorer l'espace de recherche sans être piégé par les optima locaux. À mesure que la température décroît, le recuit est de moins en moins capable d'explorer l'espace de recherche et la recherche se concentre sur une zone déterminée : l'exploitation prend le dessus sur l'exploration. Par conséquent la vitesse de décroissance de la température détermine la vitesse à laquelle le recuit « converge » vers une solution.

3.5 Recherche tabou

Tout comme le recuit simulé, la recherche tabou est dotée d'un mécanisme qui lui permet de s'échapper des optima locaux. Cependant, le mécanisme utilisé est totalement différent de celui du recuit simulé. Il est basé sur la constatation suivante : lors de l'utilisation d'un *hill-climber* le chemin[8] suivi peut se recouper pour constituer un *cycle*. Dans ce cas, le *hill-climber* passe indéfiniment sur les solutions constituant le cycle. Le même phénomène peut se produire au voisinage d'un

[8] Voir définition 11, page 24

optimum local si une heuristique tente de sortir du bassin d'attraction[9] de l'optimum local. La recherche tabou permet d'éviter ce problème : dans sa version la plus élémentaire, la recherche tabou est un *hill-climber* doté d'une mémoire de mouvement et d'un mécanisme d'échappement [Glo89a, Glo89b, OK95]. La mémoire de mouvement est utilisée comme historique de la recherche pour interdire les cycles courts et permettre ainsi de « sortir des optima locaux ». Le mécanisme d'échappement (appelé *critère d'aspiration*) est utilisé pour lever l'interdiction d'utilisation d'un mouvement si ce dernier conduit à une meilleure solution.

Problème	Références
ordonnancement, planification, production	[OL96, Özd96, TCM95, HW95, HW96, NS96b, Tal98, VAL96, Glo95, Tai93b, Pen94, EAVA97, NS97, DPT96b, PD99]
conception d'emploi du temps	[OL96, Her91]
aviation, conflits ATC	
ordonnancement d'instructions, micro-code	
constitution et ordonnancement de trains	
construction de carte génétique	
routage de paquets dans un réseau	
électronique (routage,...)	[Tal98, GOD+98]
robotique	
économie, finance	[OL96]
chimie, biochimie	
reconnaissance de formes	
optimisation de réseau (gaz, fluides, télécom.)	[OL96, Tal98]
gestion de réseau de distribution électrique	
optimisation de structure mécanique	
conception de matériaux composites	
conception et apprentissage de réseau de neurones	
jeux (taquin, ...)	
optimisation de fonction numérique	[BT94]
coloriage de graphe	[OL96, Tal98]
clique maximale dans un graphe	[Tal98, Glo95, SG97b]
partitionnement de graphe	[OL96, Tal98, Glo95, GOD+98]
sac à dos, problèmes de découpe	[OL96, BT94, Tal98, Glo95, AFH95, FB98]
couverture d'ensemble, partitionnement d'ensemble	[Glo95]
applications médicales	
problème du voyageur de commerce (TSP)	[OL96, Tal98, Glo95]
problème d'affectation quadratique (QAP)	[OL96, BT94, BPT96, Tal98, Tai93b, THG97, THG98, BPT97, Bac99]
problème de satisfaisabilité (SAT)	[Tal98]
introduction, état de l'art	[Ree95, RSORS96, OL96, Tal98, HTdW97, Glo89a, Glo89b, BT94, Glo95]

Tableau 3.3: Quelques domaines d'application de la recherche tabou.

Dans cette section, nous ne présentons que la version *standard* de la recherche tabou. Il existe cependant beaucoup de variantes de cette méta-heuristique et les domaines d'application sont extrêmement nombreux (voir tableau 3.3). Les principales variantes concernent la gestion de la liste tabou, l'ajout de mémoires à court et à long terme, l'utilisation de critère d'aspiration dédié au problème traité, la parallélisation [BT94, Glo89a, Glo89b, Glo95, HTdW97].

[9] Voir section 2.3.1, page 17

3.5.1 Terminologie

Les mouvements inverses des derniers mouvements exécutés sont mémorisés dans une structure de données particulière pour interdire les cycles courts :

Définition 20 (liste tabou)
La liste tabou est une structure de données dans laquelle sont mémorisés les k mouvements inverses des k derniers mouvements effectués. k est la taille de la liste tabou.

Dans sa version la plus élémentaire, une liste tabou de taille k peut être réalisée à l'aide d'une liste circulaire de taille k.

Définition 21 (mouvement tabou)
Un mouvement tabou est un mouvement inverse mémorisé dans la liste tabou.

Dans certains cas, ce mécanisme de liste tabou interdit un mouvement qui aurait pu conduire à une solution meilleure que les solutions rencontrées jusqu'alors. C'est pourquoi la recherche tabou est dotée d'un « critère d'aspiration » qui lève le statut tabou d'un mouvement si ce dernier conduit à une solution meilleure que les solutions préalablement rencontrées :

Définition 22 (critère d'aspiration)
Le critère d'aspiration est la condition nécessaire pour qu'un mouvement tabou soit accepté comme prochain mouvement, malgré son statut « tabou ».

Considérons s^* la meilleure solution découverte depuis le début de la recherche en cours d'exécution ; le critère d'aspiration peut simplement consister à accepter le mouvement tabou de s à s' si s' est meilleure que s^* (du point de vue du coût à optimiser).

Un voisin s' de la solution courante s est dit *aspirant* si le mouvement de s à s' vérifie le critère d'aspiration.

3.5.2 Principe de fonctionnement

La recherche tabou est un algorithme de recherche locale qui est capable de sortir d'un optimum local [Glo89a, Glo89b, Glo95, Ree95, RSORS96, GM95, OL96] contrairement aux *hill-climbers*.

L'algorithme de la recherche tabou est donné en figure 3.8.

3.6 Algorithmes génétiques

L'idée consistant à appliquer les principes de la sélection naturelle aux systèmes artificiels, introduite voilà plus de trente ans, a donné naissance à différents domaines regroupés aujourd'hui sous le terme *algorithmes évolutifs* (AE) [Hol75, Gol89, Gol94, Mic94, Mit96, BHS97, OK95, Her90, SERW95, DS87, Dav91b, Mic92, Whi93b, BBM93a, BBM93b, Pre95, Tom96, Fog94b]. Dans ce mémoire nous utilisons le terme *algorithmes génétiques* (AG) pour désigner des algorithmes hybrides qui empruntent des techniques issues des différents domaines constituant les algorithmes évolutifs.

De manière à réduire la taille de cette section, nous ne présentons que la version *standard* des algorithmes génétiques. Il existe cependant bien d'autres possibilités et les domaines d'application sont extrêmement nombreux (voir tableau 3.4).

Problème	Références
ordonnancement, planification, production	[HW96, CC88, CPP95, RC94, YR97, FRC94, CT94, HMW90, TN92, NY92, BMK96, NS96a, YN96, IHI96, Dav85, CS89, WSF89, SP91, HM91, BUMK91, Bru93, Dav91b, LSS93, FRC93, DYN93, KOY95, Chu95c, Fan92, Fan94, DPT95a, DPT96b, Por96, SF96, Tal98, DP92, Soa94, SF97, FB91, UBKM93, Hus93, Ghe95, PG95, OL96, FF95, KBH94a, VAL96, CP96, Mic92, DG94, CM91, Jul93, Hus94, WSS91, NDY94, SDMW96, FM91, CS89, NY91, Tai93b, Pen94, WFMS94, EM93, Kid93, FVC95, Evo97, Par92, TCM95, MV94, Gil97, LGWFP97, RVLS97, RHCC97, IMT97, DP95, DPT95c, DPT95b, COC97, TS97, LRS+98b, HR98, LR97]
conception d'emploi du temps	[RC94, RCF94b, AA92, CDM90, BEW95, Tal98, RCF94a, OL96, Mic92, Fan92, Fan94, Pae94, Evo97, CRF94, Ngu97, Ven97, PRCF98]
aviation, conflits ATC	[MDA94, vKHHK95, AAK+93, Tal98, DASF94, OD98]
ordonnancement d'instructions, micro-code	[BWJ90]
constitution et ordonnancement de trains	[vWKvdBvK94, GBH+91]
construction de carte génétique	[GS97]
routage de paquets dans un réseau	[SSW93, OL96, Kid93]
électronique (routage, ...)	[RC94, HK94, OL96, Tal98, RO93, Hul90, SV95, LC95b, MK93, BM95, Hul97, RAARS98, GOD+98]
robotique	[RC94, BATM93, Mic96, TBAM, JGSB92]
économie, finance	[SAS92, OL96, Gil97]
chimie, biochimie	[SK92, OL96, Tal98, PS93, Yur94, McC97, HKB94, RAT97]
reconnaissance des formes	[Gol94, Tal98]
optimisation de réseau (gaz, fluides, télécom)	[RC94, Gol94, OD93, OL96, MH93b, KCR95, CD87, BWM97, TMK97, DAS97, CM98, VHS98, Cal99]
gestion de réseau de distribution électrique	[MDRS96]
optimisation de structure mécanique	[FF93, MDRS96, Gol94, Tal98, PS93, SX93, RKLH95, RP96]
conception de matériaux composites	[MDRS96, RKLH95]
réseau de neurones (conception/apprentissage)	[OFM92, Gru94, PKS93, Tal98]
jeux (taquin, ...)	[SL90, Mic92, Yun97]
optimisation de fonctions numériques	[Sch85, Dav91a, Gol94, OL96, Mic95, MA94, Tal98, CP96, BBM93d, Mic92, TF93, GW93, ZK, RVSK97, JM91a, Khu94, PS93, SX93, Bäc94, Rud94b, Yur94, Boo87, Ack87, ES91, RP96, KP98, KM98, LRS98a, OTT98, KF98, Mar94, VRUC97, LR97]
coloriage de graphe	[EvdH97, FF95, OD93, Tal98, DH98, RH98, Cal99]
clique maximale dans un graphe	[FF95, OL96, Tal98]
partitionnement de graphe	[FF95, KBH94a, TBAM, Tal98, Mic92, vLM90, BM95, DPT95c, GOD+98]
sac à dos, problèmes de découpe	[KBH94b, KSV92, OL96, Tal98, Mic96, GW93, KSV90, IKB+97, AFH95, PK94, RK98, ZT98]
couverture/partitionnement d'ensemble	[RPLH89, OL96, Tal98, Lev94, Lev93, JB91, Lev93]
applications médicales	[PvGvL+92, LK92, Gol94, WFMS94, Yur94]
voyageur de commerce (TSP)	[WSS91, JSG89, WSF89, OL96, FF95, Tal98, Mic92, Mic96, SW87, SDMW96, AM95, MTS93, TF93, Yur94, Gre87b, CM95b, CPP95, LRS+98b]
affectation quadratique (QAP)	[BPT96, OL96, FF95, Tal98, Tai93b, DPT95c, Mar94, BPT97, Bac99]
satisfaisabilité (SAT)	[GV97, DH94, Par95, Mic96, OL96, FF95, Tal98, EvHMS98]
introduction, état de l'art	[Hol92b, Gol89, Gol94, Mic92, Dav87, Ste92, DM92, Mit96, Dav91b, Ree95, RSORS96, OL96, Tal98, Tom96, FF95, Mic96, Hol92a, Rio92, BBM93a, BBM93b, RC94, Whi93b, Pre95, Duv94, Tan89, Whi93a, ZK93, BD93, Ven97, CPP95, Ada94]
modélisation/étude des AG	[De 95, Aga98, FL99, Kal98, MF97, Par93, QP94, WY94, Sha98, Pre94a, Pre94b, HB91, Gre93, Gol87, ER96, Raw91, Whi93c, WV94]

Tableau 3.4: Quelques références sur les AG.

```
Créer une solution initiale s
Évaluer s
Mémoriser la meilleure solution trouvée s*←s
Initialiser la liste-tabou (LT)
Répéter ...
   Choisir le meilleur voisin s' de s dans V_O(s)
           tel que m(s,s') ∉ LT ou s' est aspirant
   s'←s
   Mettre à jour la liste-tabou (LT)
   Si f(s') meilleure que f(s*) alors
      s*←s'
Tant que critère-d'arrêt non satisfait
Afficher("Solution trouvée " s*)
```

Figure 3.8: Principe de fonctionnement de la recherche tabou.
La meilleure solution trouvée est notée s^*. Le mouvement de s à s' est noté $m(s, s')$.

3.6.1 Terminologie

Nous introduisons en ce point la terminologie communément utilisée dans le domaine des algorithmes évolutifs. Une solution potentielle s est appelée un *individu*. La fonction coût f, est désignée sous le terme *fonction objectif*. Elle détermine le degré de validité, ou encore le degré d'adaptation, d'un individu comme solution au problème.

Il est nécessaire d'introduire un certain nombre de termes avant d'aborder plus précisément les algorithmes génétiques. Par conséquent, nous définissons les termes suivants :

Définition 23 (génotype)
Le génotype est la structure de données utilisée pour coder un individu.

Définition 24 (phénotype)
Le phénotype est l'interprétation du génotype en tant que solution au problème traité (décodage du génotype).

Définition 25 (population)
Une population est un groupe d'individus sur lequel l'AG effectue un certain nombre d'opérations.

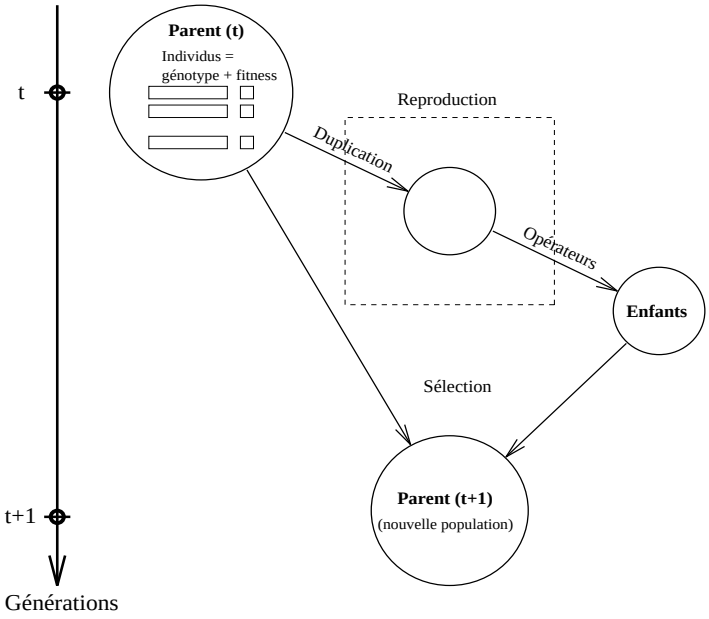
Par analogie avec la génétique, un *individu* est constitué d'un ou plusieurs *chromosomes*. Un *gène* est une partie d'un chromosome. Un *allèle* est la *valeur* d'un gène. Pour *l'algorithme génétique standard*, les chromosomes sont des chaînes de bits de longueur l. Les chromosomes sont des éléments de l'espace de recherche $\mathcal{E} = \{0, 1\}^l$. Les gènes sont par conséquent des bits et les allèles des valeurs binaires. Nous dirons qu'un gène a *convergé* lorsque 95% de la population possède la même valeur pour ce gène. Nous considérons que la population a convergé lorsque tous les gènes ont convergé.

L'algorithme génétique manipule une population d'individus qui évolue au cours du temps grâce aux divers opérateurs (crossover, mutation, sélection) en favorisant la reproduction des individus performants. Il est efficace tant que les individus de la population sont différents. Dès qu'ils deviennent *quasi-identiques*[10] l'algorithme ne parvient plus à réduire le coût des solutions (autrement dit à améliorer le degré d'adaptation des individus). Pour éviter de laisser s'exécuter l'algorithme

[10]Nombre élevé d'allèles communs entre les individus.

jusqu'à ce point, une mesure de l'hétérogénéité de la population d'individus est définie sur la population : cette mesure est appelée *diversité* de la population. À l'instar de [FF95], la diversité peut être mesurée en terme d'entropie. Une méthode plus simple (mais moins précise) consiste à effectuer une mesure d'écart-type sur les coûts des individus de la population.

3.6.2 Principe de fonctionnement



(a) Principe de fonctionnement d'un AG.

```
Créer une population initiale
Calculer le coût des individus de la population
Répéter ...
  Reproduction
    Sélectionner les candidats à la reproduction
    Dupliquer les individus
  Appliquer les opérateurs génétiques
  Évaluation
    Calculer le coût des nouveaux individus
  Sélection
    Sélectionner les survivants
                parmi les parents et les enfants
    Constituer la nouvelle population
Tant que critère-d'arrêt non satisfait
Afficher("Solution trouvée "
         meilleur_individu_de_la_population)
```

(b) Pseudo-algorithme d'un AG.

Figure 3.9: Algorithme génétique *standard*. Les étapes sont stochastiques.

Le principe de fonctionnement d'un algorithme génétique standard est représenté sur la figure 3.9. L'AG est initialisé à partir de N individus (appelés *population initiale*) générés aléatoirement ou à partir d'une méthode dédiée au problème traité. Le résultat de l'application de la fonction d'évaluation (*i.e.* le calcul du coût) mesure le degré d'aptitude de l'individu à résoudre le problème posé. Une fonction de sélection est alors appliquée afin de favoriser au cours du temps les individus les mieux adaptés, à les faire se reproduire (duplication) et à appliquer sur leur descendance des opérateurs génétiques pour engendrer à nouveau une *population* P constituée de ces nouveaux individus. Une fraction de la population est remplacée par sa descendance à chaque *génération* t pour donner naissance à la population $P(t+1)$. L'*écart entre les générations* ou *taux de remplacement* (*generation gap*) est un nombre compris entre 0 et 1, il indique le pourcentage de parents remplacés par des enfants. Le processus est itéré et, au cours des générations, la population converge vers une certaine classe d'individus qui représentent des approximations de la solution au problème initialement posé. Ce processus est stoppé au bout d'un nombre fixé de générations, ou lorsque les individus ont convergé vers une ou plusieurs solutions satisfaisantes. Le meilleur individu de la population est alors retenu comme solution au problème posé.

Afin de compléter cette présentation de l'algorithme génétique standard, nous reprenons en détail certaines de ses étapes. Après la description de l'étape d'évaluation des individus, nous évoquerons rapidement les opérateurs génétiques classiquement utilisés. Nous terminerons cette description du

principe de fonctionnement de l'algorithme génétique standard par un court exposé de la phase de sélection.

Évaluation des individus Cette étape consiste à évaluer les individus de la population. Le nombre d'enfants engendrés[11] par un individu sera biaisé par son degré d'adaptation. Les individus les mieux adaptés tendent à fournir la descendance la plus nombreuse.

Les opérateurs génétiques Après évaluation et reproduction, les opérateurs génétiques agissent sur les individus de la nouvelle population constituée de la progéniture de la population précédente.

(a) Crossover. (b) Mutation.

Figure 3.10: Exemple d'opérateur de recombinaison (crossover) et de mutation.
Partant de deux solutions initiales (individus parents) l'opérateur de recombinaison engendre deux nouvelles solutions (enfants) en échangeant des informations des solutions initiales dans les nouvelles solutions.
L'opérateur de mutation modifie une information élémentaire (gène) dans une solution pour engendrer une nouvelle solution potentielle.

Le premier opérateur est l'opérateur de recombinaison, également appelé *crossover*. Il agit sur deux individus en s'inspirant de la reproduction sexuée où le génotype d'un enfant est obtenu par recombinaison des génotypes des deux parents (voir figure 3.10(a)). Cet opérateur est classiquement considéré comme l'opérateur le plus important dans les algorithmes génétiques. Dans sa version « standard », le principe de cet opérateur consiste à définir un point de coupure et à échanger les gènes entre les individus de part et d'autre de ce point.

Le second opérateur est la *mutation*. Dans sa version « standard », il modifie de manière aléatoire la valeur d'un ou plusieurs gènes (voir figure 3.10(b)).

La présence de plusieurs opérateurs dans l'AG permet de définir des variantes en fonction de la méthode utilisée pour appliquer les opérateurs. Nous appelons *schéma d'application des opérateurs* la représentation schématique illustrant le mode d'utilisation des opérateurs appliqués aux individus-parents pour engendrer les individus-enfants. Pour chaque parent, cette représentation fait apparaître l'ordre dans lequel sont appliqués les opérateurs.

Le schéma d'application des opérateurs utilisé dans un algorithme génétique standard est représenté sur la figure 3.11.

Sélection Cette étape constitue la population de la génération suivante à partir des parents et des enfants de la génération courante. Une fraction de la population est remplacée par sa descendance à chaque *génération*. L'écart entre les générations indique la proportion de parents remplacés par des enfants. Le choix des individus conservés est généralement basé sur leur degré d'adaptation. Par exemple, pour un problème de maximisation du coût, la *méthode de la roulette* attribue une probabilité de sélection d'un individu x proportionnelle à $\frac{f(x)}{\overline{f}}$ ($f(x)$ est le coût de l'individu x, $\overline{f}$ est la

[11] par duplication

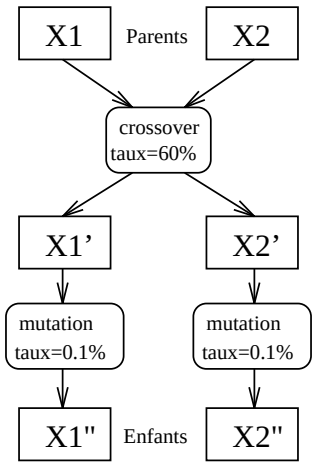


Figure 3.11: Schéma d'application classique des opérateurs : deux individus sont sélectionnés dans la population ; l'opérateur de recombinaison (crossover) est appliqué, avec une certaine probabilité, sur ces deux individus. Il engendre deux individus sur lesquels l'opérateur de mutation est appliqué.

moyenne des coûts des individus de la population). Certaines stratégies prennent systématiquement les meilleurs, d'autres en revanche utilisent le degré d'adaptation comme biais pour la sélection.

La sélection des survivants étant généralement stochastique, le meilleur individu d'une génération peut ne pas survivre, d'où l'introduction de stratégies élitistes. L'élitisme consiste à conserver l'individu le plus adapté dans la population d'une génération sur l'autre. Il a été montré qu'un algorithme génétique ne converge jamais vers un optimum global s'il n'utilise pas une stratégie élitiste [Rud94a].

3.6.2.1 Algorithmes génétiques générationnel et stationnaire

Deux types distincts d'algorithmes génétiques sont définis : les algorithmes génétiques générationnels et les algorithmes génétiques stationnaires[12]. Les premiers renouvellent complètement la population à chaque génération. Les seconds introduisent un nouveau couple d'individus dans la population à chaque génération.

3.7 Autres méthodes

Dans les précédentes sections, nous avons présenté trois méta-heuristiques basées sur une méthode amélioratrice. Elles font partie d'un ensemble de méthodes de recherche largement répandues parmi lesquelles figurent :

- les systèmes multi-agent [SV97, BLGR99];
- les systèmes experts ;
- la recherche dispersée (*scatter search*) [Glo97, OL96, CMMT97, CMMT98] ;
- les méthodes de satisfaction de contraintes, programmation par contrainte [OL96, YGMTH94, Rod94, MPT+99] ;

[12]Conformément à [FF95] nous utilisons ce terme comme traduction de *steady-state*

- les méthodes d'énumération implicite (procédures par séparation et évaluation/*branch and bound*, ...) [AF90, Lév94, GM95, Ree95, RSORS96, PS82, CM95a, Sun95, SD96, RH96, CN96, Chu95a, Chu92b, Chu92a] ;
- les colonies de fourmis [OL96, DG96, DG95, DG97, Rou98, RFRT98, TRFR99, Tal98, CDMT94, SG97a, TG97] ;
- les réseaux de neurones [DN92, Hop88, Ree95, RSORS96, Gru94, OK95, OL96, SD90, Tal98, PvGvL$^+$92, VAL96, SH87] ;
- la programmation dynamique [CLR94, CC88, CP96, EL99, GM95, Ree95, RSORS96, PS82] ;
- méthode du simplexe, programmation linéaire, programmation non linéaire [dD93, PS82, AF90, Lév94, Sun95, Kor71, SG80, Van78] ;

Compte tenu de la diversité des méta-heuristiques, et plus généralement des méthodes de recherche, nous n'entrerons pas davantage dans les détails. Dans cette section, nous présentons rapidement quelques méthodes fréquemment utilisées pour engendrer les solutions initiales des méta-heuristiques : les méthodes approchées.

En général, lorsqu'il s'agit de résoudre un problème concret, le fait de disposer rapidement d'une solution quasi-optimale est plus que satisfaisant. Les algorithmes d'approximation génèrent des solutions réalisables quasi-optimales en un temps raisonnable [CLR94, GM95]. Il existe des algorithmes d'approximation pour lesquels il est possible de calculer une borne :

Définition 26 (borne pour les algorithmes d'approximation)
Pour un problème donné, un algorithme d'approximation a une borne $\rho(n)$ si pour toute instance de taille n, le coût C de la solution générée par l'algorithme est éloigné au maximum d'un facteur $\rho(n)$ du coût C^* d'une solution optimale :

$$\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq \rho(n)$$

Il s'avère parfois plus commode d'utiliser une mesure de l'erreur relative de l'algorithme d'approximation. Cette mesure est définie par $\frac{|C-C^*|}{C^*}$, elle permet de définir la borne d'erreur relative d'un algorithme d'approximation :

Définition 27 (borne d'erreur relative)
Pour un problème donné, un algorithme d'approximation a une borne d'erreur relative $\varepsilon(n)$ si pour toute instance de taille n :

$$\max\left(\frac{C}{C^*}, \frac{|C-C^*|}{C^*}\right) \leq \varepsilon(n)$$

Comme précédemment, C représente le coût de la solution générée par l'algorithme, et C^* est le coût d'une solution optimale.

Ces bornes sont liées par les relations suivantes : pour un problème de minimisation $\varepsilon(n) = \rho(n) - 1$, pour un problème de maximisation $\varepsilon(n) = \frac{\rho(n)-1}{\rho(n)}$.

Pour certains problèmes, il existe des algorithmes d'approximation dont les bornes d'erreur relative diminuent en fonction du temps de calcul utilisé :

Définition 28 (schéma d'approximation)
Pour un problème donné, un schéma d'approximation est un algorithme d'approximation dont les entrées sont une instance du problème et une valeur $\epsilon > 0$, tel que pour tout ϵ fixé à l'avance

l'algorithme se comporte comme un algorithme d'approximation dont la borne d'erreur relative est égale à ϵ.

On dit qu'un schéma d'approximation est un schéma d'approximation en temps polynomial si pour tout $\epsilon > 0$ fixé, l'algorithme s'exécute en temps polynomial par rapport à la taille n de l'instance considérée (en entrée).

Lors de la résolution d'un problème concret, il est assez rare de disposer de la valeur du coût d'une solution optimale. De même, il n'est pas toujours possible de calculer la borne des algorithmes d'approximation utilisables pour le problème traité. Dans ce cas, le calcul de bornes inférieures et supérieures aussi proches que possible de la valeur optimale est la seule méthode utilisable pour évaluer la qualité des solutions obtenues. Même s'il n'est pas possible de situer le coût des solutions engendrées par rapport au coût d'une solution optimale, ces bornes peuvent être utilisées en entrée d'une méthode de recherche. Elles peuvent également montrer l'optimalité d'une solution lorsque les bornes inférieures et supérieures ont la même valeur.

Les méthodes utilisées pour générer des bornes supérieures sont très variées, il peut s'agir par exemple d'une méta-heuristique ou d'un algorithme glouton. Nous allons présenter deux techniques fréquemment utilisées pour concevoir des algorithmes d'approximation : les algorithmes gloutons et les méthodes de relaxation.

3.7.1 Algorithmes gloutons

Les algorithmes gloutons sont très fréquemment utilisés pour initialiser une méta-heuristique, ou plus simplement pour obtenir rapidement une solution acceptable [Ree95, Lév94, GM95, RSORS96, Glo86, CLR94, OK95, PS82, HH97]

Un algorithme glouton effectue, à chacune de ses étapes, un choix qui semble le meilleur (en terme de coût) au moment où ce choix est effectué. L'algorithme ne revient jamais sur un choix précédemment effectué.

Il existe de nombreux algorithmes de ce type. Prenons un exemple dans le domaine de l'ordonnancement : l'objectif consiste à ordonnancer un ensemble d'opérations sur un ensemble de machines en vue de minimiser la durée globale de production. Un algorithme glouton extrêmement simple consiste à choisir à chaque étape l'opération de durée minimale non encore ordonnancée (en respectant les éventuelles contraintes imposées par le problème d'ordonnancement).

3.7.2 Méthodes de relaxation

Les méthodes *de relaxation* fournissent une solution approchée au problème traité. Elle sont en général conçues de manière à ce que la solution obtenue puisse être située (en terme de coût) par rapport à la valeur optimale. De telles méthodes permettent d'obtenir des bornes inférieures ou supérieures de la valeur optimale du coût. Par exemple, la relaxation lagrangienne [GM95, TCM95, Ree95, RSORS96, Glo86, PS82, CCP94] fait partie des méthodes largement utilisées.

3.8 Conclusion

Bien que non exhaustif, cet état de l'art présente les principales méta-heuristiques basées sur une méthode amélioratrice. Ces méta-heuristiques, largement utilisées dans tous les domaines, figurent parmi les plus performantes à l'heure actuelle. Cependant les meilleures performances sont obtenues par des méthodes hybrides alliant plusieurs méthodes de recherche [FF95, OK95, Tal98, OL96]. Dans le prochain chapitre, nous présentons les techniques d'hybridation.

Chapitre 4

Méthodes hybrides

Après une définition des méthodes hybrides, ce chapitre présente les principales techniques d'hybridation, il apporte des réponses à deux questions clefs :
- quelles méthodes hybrider ?
- comment hybrider ?

4.1 Introduction

Les méta-heuristiques sont applicables à un grand nombre de problèmes sans nécessiter de connaissances approfondies sur le problème traité ou sur les méta-heuristiques. Elles permettent d'obtenir de bonnes performances en moyenne, mais sur certains problèmes, ou certaines instances d'un problème, les performances sont parfois médiocres. À l'inverse, il existe des méthodes dédiées qui permettent d'obtenir d'excellents résultats pour certains problèmes, ou certaines instances particulières. Malheureusement ces méthodes sont bien souvent difficilement applicables à plusieurs problèmes car elles exploitent toutes les spécificités du problème pour lequel elles ont été conçues.

Une « bonne » méthode de recherche ne doit pas uniquement être applicable à un grand nombre de problèmes, elle doit également être performante (en terme de robustesse et de qualité de solution finale). Pour être performante, une méthode de recherche doit associer judicieusement exploration et exploitation de l'espace de recherche. Or une méthode de recherche est rarement aussi efficace pour exploiter que pour explorer l'espace de recherche. Une solution consiste à ajouter des mécanismes complémentaires dans une méthode de recherche donnée. Cependant cette opération peut s'avérer assez délicate et provoquer des effets de bord nuisibles au bon fonctionnement de la méthode de recherche. De ce fait, il peut être extrêmement bénéfique d'associer une méthode de recherche[1] dont la capacité d'exploration[2] est très élevée à une méthode de recherche[1] dont le point fort est l'exploitation[2] de l'espace de recherche. Par ailleurs, une (seule) méthode ne permet pas d'obtenir de très bons résultats sur un large ensemble de problèmes [WM95, WM97] (voir section 2.3.2, page 18). En ce cas pourquoi se limiter à l'utilisation d'une (seule) méthode alors qu'il serait possible d'utiliser (simultanément) plusieurs méthodes de recherche pour améliorer les performances ? D'où l'idée de fédérer ces méthodes de recherche en une méthode hybride :

Définition 29 (méthode hybride)
Une méthode hybride est une méthode de recherche constituée d'au moins deux méthodes de recherche distinctes.

En résumé, que ce soit pour élargir le spectre d'application d'une méthode de recherche ou pour augmenter ses performances, l'hybridation est une technique qui peut s'avérer extrêmement efficace.

[1] ou plusieurs
[2] Voir section 2.3, page 15

L'hybridation n'offre pas que des avantages, ainsi selon le type de couplage entre les méthodes hybridées, il sera peut être nécessaire de modifier chacune des méthodes, ne serait-ce que pour leur permettre d'échanger des informations dans un format commun. Un autre problème induit par l'hybridation est la multiplication des paramètres : chaque méthode de recherche possède ses propres paramètres qu'il s'agit d'ajuster au mieux pour obtenir de bonnes performances. L'utilisation d'un modèle parallèle ou l'autoadaptation[3] sont des techniques qui permettent de s'affranchir de ce type de problème : par exemple, [FRP97b] mettent en œuvre un algorithme génétique parallèle utilisant simultanément plusieurs paramétrages distincts. De plus, les méthodes de recherche sont en général assez robustes vis-à-vis du paramétrage, ce qui facilite la résolution de ce problème.

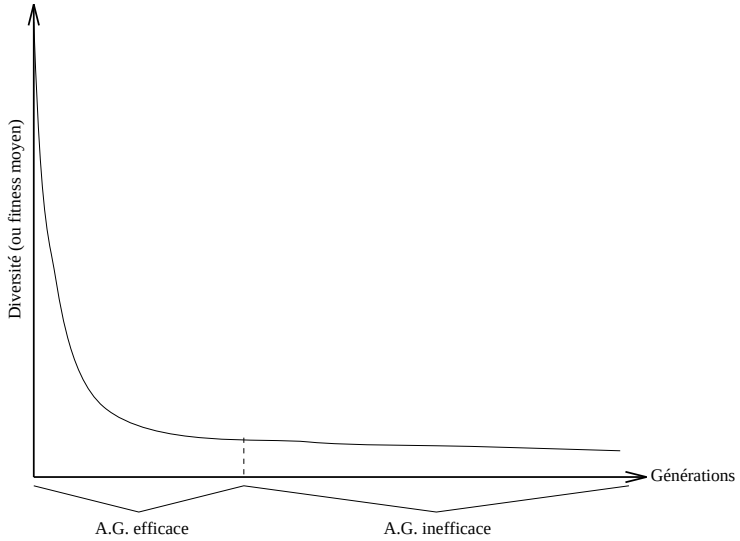


Figure 4.1: Évolution de la diversité (ou de la moyenne du coût des individus) de la population d'un AG au cours du temps lors de la résolution d'un problème de minimisation du coût. L'AG est efficace tant que les individus constituant la population sont différents. Durant cette phase l'AG explore efficacement l'espace. Dès que la diversité chute, l'AG ne parvient plus à réduire le coût des solutions. Les individus ont convergé vers une zone de l'espace. L'AG devient alors inefficace pour exploiter cette zone.

Nous avons vu précédemment que pour définir une méthode hybride efficace, il faut savoir caractériser les points forts (et les points faibles) de chaque méthode de recherche. Par exemple, les algorithmes génétiques sont très performants lorsqu'il s'agit d'explorer l'espace de recherche, mais ils s'avèrent ensuite incapables d'exploiter efficacement la zone vers laquelle la population converge. Il est alors bien plus intéressant (à la fois en terme de durée d'exécution et de qualité de solution) de stopper l'algorithme génétique pour utiliser une autre méthode (voir figure 4.1).

L'hybridation peut également être utilisée pour résoudre simultanément différents aspects d'un même problème : cette méthode est fréquemment utilisée dans le domaine de la gestion de production [Özd96, HH97, HW96] où se posent simultanément différents problèmes tels que les affectations de machines et de personnel à des opérations, la gestion des stocks ...

L'hybridation n'est pas la seule possibilité à envisager : une alternative à l'hybridation consiste à relancer la méthode de recherche utilisée avec une autre solution initiale et/ou un autre paramétrage dès que cette dernière ne parvient plus à améliorer le coût de la solution courante.

[3]Dans le cas d'un paramétrage *autoadaptatif* la méthode gère elle-même son paramétrage en fonction de son évolution au cours du temps [SS94, Bäc92, Spe91, Gre86, SP91, CM91, DBB94, Sin98, Edm98, TR98, Boo87].

4.2 Quelles méthodes hybrider ?

Il est possible d'hybrider toutes les méthodes, y-compris méthode exacte et heuristique [DP95, CL95a, LC95a, MM98]. Pratiquement, le souci de performance, ou les contraintes de ressources informatiques limitent les possibilités d'hybridation. Donc, si *a priori* il est possible d'hybrider n'importe quelles méthodes ; en pratique il faut être prudent sur le choix des méthodes utilisées pour obtenir une bonne coopération entre les constituants de la méthode hybride.

4.3 Comment hybrider ?

Nous nous appuyons sur les résultats de plusieurs travaux effectués au sein du groupe de travail PERFORM[4] pour exposer les principales techniques d'hybridation [DPT95c, PT95a, PT95b]. L'hybridation peut prendre place à deux niveaux : soit plusieurs méthodes de recherche coopèrent au sein d'une méthode hybride, soit une heuristique (ou une méthode de recherche) est insérée à l'intérieur d'une autre méthode de recherche. Dans ce dernier cas, il est nécessaire de définir à quel niveau, c'est-à-dire dans quel composant de la méthode de recherche, va prendre place l'hybridation. Contrairement aux versions « standard » des méthodes de recherche où le seul lien avec le problème traité est la fonction coût, il est possible d'insérer des heuristiques[5] dans chaque composant de base d'une méthode hybride en utilisant :

- un codage des solutions dédié au problème ;
- une heuristique dédiée pour créer la (ou les) solution(s) initiale(s) ;
- des critères liés au problème traité dans la fonction coût ;
- des opérateurs dédiés (éventuellement remplacés par une/plusieurs méthode(s) de recherche) ;
- un schéma d'application des opérateurs non traditionnel ;
- un paramétrage adapté au problème traité (paramétrage adaptatif, ...).

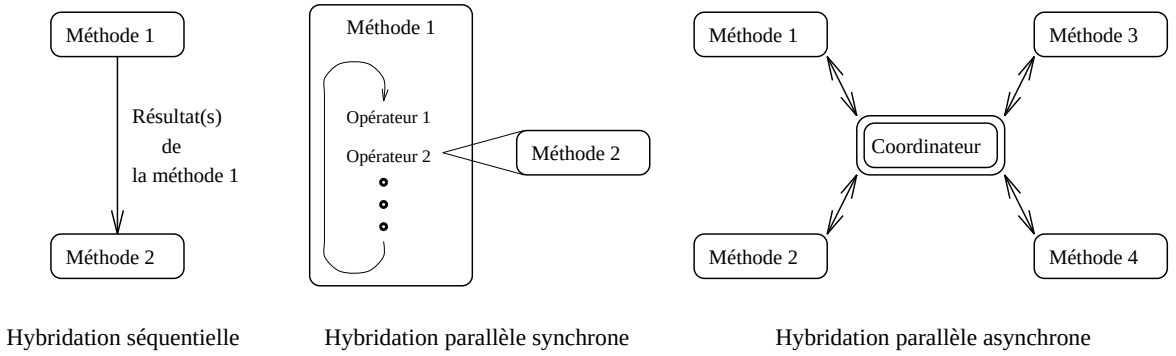


Figure 4.2: Techniques d'hybridation.

Comme le montre le paragraphe précédent, l'hybridation peut prendre place dans un ou plusieurs composants d'une méthode de recherche. Elle peut également consister à assembler plusieurs méthodes de recherche au sein d'une méthode hybride ... En première approche, les différentes techniques d'hybridation peuvent être réparties en trois catégories principales [PT95a, PT95b] (voir figure 4.2) :

[4]PERFORM (*Parallel gEnetic algoRithms FOR optiMization*) est un groupe de travail qui rassemble des chercheurs du Laboratoire d'Informatique Fondamentale de Lille (LIFL) et du Laboratoire d'Informatique du Littoral (LIL). Il est possible d'accéder à ce groupe de travail sur Internet via l'adresse `http://www.lifl.fr/~talbi/perform.html`.

[5]ou plus généralement des méthodes dédiées.

- hybridation séquentielle ;
- hybridation parallèle synchrone ;
- hybridation parallèle asynchrone.

L'hybridation *séquentielle* consiste à exécuter séquentiellement différentes méthodes de recherche de telle manière que le (ou les) résultat(s) d'une méthode serve(nt) de solution(s) initiale(s) à la suivante. Cette technique d'hybridation est la plus simple, elle ne nécessite pas de modification des méthodes de recherche utilisées : il suffit de pouvoir initialiser chaque méthode de recherche à partir de solutions pré-calculées.

L'hybridation *parallèle synchrone* est obtenue par incorporation d'une méthode de recherche particulière dans un opérateur. C'est – en quelque sorte – une hybridation plus fine que la précédente étant donné que l'une des méthodes de recherche est englobée dans l'autre. Cette technique est plus complexe à mettre en œuvre que la précédente. Il faut tenir compte des fortes interactions entre les méthodes incorporées dans ce type d'hybridation, lors du choix de ces méthodes, pour ne pas aboutir à une méthode hybride moins efficace (en terme de coût de solution finale) que les méthodes qui la constituent.

L'hybridation *parallèle asynchrone* consiste à faire évoluer en parallèle différentes méthodes de recherche. Cette coévolution permet une bonne coopération des méthodes de recherche au travers d'un coordinateur : chargé d'assurer les échanges d'informations entre les méthodes de recherche constituant l'hybride, son rôle est essentiel pour une bonne coopération de l'ensemble. Cette technique d'hybridation nécessite des modifications mineures des méthodes de recherche utilisées afin d'assurer les communications avec le coordinateur. Elle offre l'avantage de faire fonctionner « de pair » les différentes méthodes de recherche et de laisser ainsi à tout moment la possibilité à chaque méthode d'exploiter les résultats des autres méthodes en vue d'une utilisation optimale de ses potentialités d'exploration/exploitation. En contrepartie, elle nécessite la réalisation d'un coordinateur.

Il est bien entendu possible de combiner plusieurs techniques d'hybridation au sein d'une méthode de recherche hybride. Dans le cadre de nos travaux, nous nous limitons à cette classification, cependant [Tal98] propose une taxinomie des méta-heuristiques hybrides et introduit une grammaire qui permet de décrire précisément les différents schémas d'hybridation.

Le tableau 4.1) présente quelques domaines où l'hybridation a permis d'améliorer les résultats.

4.4 Conclusion

Bien qu'il soit possible d'hybrider n'importe quelles méthodes entre elles, la recherche de performances, ou les ressources informatiques limitent les combinaisons effectivement utilisables.

De manière utopique, une hybridation idéale engendre une méthode hybride qui cumule les bonnes propriétés de ses constituants tout en utilisant ces mêmes constituants afin d'annihiler leurs points faibles. D'un point de vue pragmatique, une méthode hybride efficace (à la fois en terme de robustesse et de qualité de solution finale) est une méthode qui tire profit des interactions entre ses constituants pour maintenir un compromis entre exploration et exploitation.

Quel que soit le problème, une hybridation efficace améliore toujours les résultats obtenus par des méthodes « non-hybrides » à la fois en terme de coût de solution finale et de temps de calcul.

Dans le prochain chapitre, nous définissons le problème sur lequel nous allons évaluer différentes méthodes hybrides : le *jobshop*.

Problème	Références
ordonnancement, planification, production	[Özd96, Bra96, Tal98, HH97, FRC94, DPT96a, CL95a, LC95a, DPT95c, NS96a, NS96b, YR97, OL96, Fan94, YN96, UBKM93, WFMS94, EM93, Kid93, PG95, Evo97, Par92, HW96, Por96, AC94, RVLS97, RHCC97, IMT97, DP95, GPP95, DPT96b, MPT+99, HR98]
conception d'emploi du temps	[Tal98, Lin92, BEW95, OL96, Evo97]
aviation, conflits ATC	[Tal98, vKHHK95, MDA94, OL96, AAK+93, DASF94]
ordonnancement d'instructions (micro-code)	
constitution et ordonnancement de trains	
construction de cartes génétiques	
routage de paquets dans un réseau	[Sin98, Kid93]
électronique (routage, ...)	[Tal98, OL96, SV95, LC95b, GOD+98]
robotique	[TBAM, BATM93]
économie	
chimie, biochimie	[Tal98, Yur94]
reconnaissance de formes	[Tal98, OL96]
optimisation de réseaux (gaz, fluides, télécom.)	[Tal98, OL96, Sin98, AC94, Cal99]
gestion de réseau de distribution électrique	[Tal98, MDRS96]
optimisation de structures mécaniques	[Tal98, FF93, MDRS96, WS94]
conception et apprentissage de réseaux de neurones	[Tal98, OL96]
jeux (taquin, ...)	[SG87]
optimisation de fonctions numériques	[Tal98, DP96, MSB91, MA94, OL96, Bäc94, Yur94, HLM98]
partitionnement de graphe	[Tal98, TMS94, DPT95c, MO96, OL96, FF95, GOD+98]
coloriage de graphe	[Tal98, OL96, FF95, DH98, Cal99]
clique maximale dans un graphe	[Tal98, BE95, OL96, FF95]
sac à dos (unidimensionnel ou multidimensionnel)	[Tal98, OL96, ML97]
couverture d'ensemble, partitionnement d'ensemble	[Tal98, OL96, Lev94, Lev93]
applications médicales	[WFMS94, Yur94]
problème du voyageur de commerce (TSP)	[Tal98, KKN93, SG87, JSG89, LPQ97, MO96, OL96, FF95, SG97a, Yur94, Gre87b]
problème d'affectation quadratique (QAP)	[Tal98, BPT96, DPT95c, OL96, Bac95, FF95, SG97a, TG97, BHPT98, BPT97, Bac99]
problème de satisfaisabilité (SAT)	[Tal98, FF95]
divers	[MG92b, MG92a, BF94, SSR94, KRC95, DBB94, MDRS96, LS94, BWJ90, MPGL93, TVG93, Pro94, CF94, BD95, MST97, AC94, Mar97, TEM95, BCL+93]
introduction, état de l'art	[Ree95, RSORS96, OL96, Tal98, FF95, Mic96, RC94, Bac95]

Tableau 4.1: Quelques domaines d'application de méthodes hybrides.

Partie II

Problème traité

Chapitre 5

Problème du jobshop

Nous traitons les problèmes d'ordonnancement car ils sont difficiles en général et d'une grande utilité pratique. Parmi ces problèmes, nous avons choisi de résoudre le problème du *jobshop* simple (JSP) pour comparer les performances de nos algorithmes car il offre un bon compromis entre difficulté de résolution et facilité de modélisation : il s'agit d'un problème $\mathcal{NP}$-dur [GJ79], très difficile à résoudre pratiquement, pour lequel de nombreuses modélisations existent.

La réalisation d'un projet suppose l'exécution préalable de multiples opérations soumises à de nombreuses contraintes. Un problème d'ordonnancement se pose lorsque le projet est décomposé en opérations interdépendantes. Résoudre un problème d'ordonnancement consiste à déterminer l'ordre et le « calendrier » d'exécution de ces opérations en leur attribuant des ressources et des dates de début, de manière à réaliser le projet.

Plus particulièrement, un problème de type *jobshop* est une modélisation d'une unité de production disposant de moyens polyvalents utilisés suivant des séquences différentes en fonction des produits. L'objectif consiste à ordonner la réalisation des produits de manière à minimiser le coût global, en respectant un certain nombre de contraintes sur les machines utilisées pour effectuer chaque opération élémentaire entrant dans la fabrication des différents produits.

Dans ce chapitre, nous évoquons rapidement les principaux problèmes d'ordonnancement ainsi que la terminologie associée au domaine. Dans un second temps, après avoir défini précisément le problème du *jobshop* simple, nous étudions ses principales modélisations.

5.1 Principaux problèmes d'ordonnancement

Il existe de très nombreuses variantes, surtout lorsqu'il s'agit de traiter des problèmes particuliers issus de la modélisation d'une situation réelle en entreprise [IHI96, NS96a, Hus94, CC88, JM74, GOT93, CP96, EL99, ST94]. Les principales variantes sont liées au type de ressources (renouvelables, non renouvelables), à la topologie des ressources (utilisées parallèlement, en série, ou capables de réaliser simultanément plusieurs opérations), et au type de production : unitaire (problèmes acycliques), en série ou par lots (problèmes cycliques).

Nous nous intéressons plus particulièrement aux problèmes d'atelier ; où les ressources utilisées sont des machines et les tâches à ordonnancer des opérations à réaliser sur ces machines. Les principaux problèmes d'atelier sont de type *flowshop*, *jobshop*, *openshop*, ou des problèmes à contraintes cumulatives.

5.1.1 Fonctions coût

La fonction coût peut tenir compte de différents critères parmi lesquels :

- flexibilité, tolérance aux pannes, vitesse de réordonnancement [FRC93, SF96, AS94] ;
- délais de fabrication, avances, retards [Bru93, CT94, CS89, HH97, IHI96, GOT93, ST94, Chu95c, BDF97, WSL95, Chu95b] ;
- durée globale du projet [YN96, IHI96, GOT93, ST94, Chu95c] ;
- charge des machines, équilibrage de charge [BUMK91, SF96, NA94, HD90, IHI96, AS94] ;
- flux moyen, temps de séjour [ST94, HH97, IHI96, GS78, WSL95] ;
- nombre de machines actives [CT94, Özd96] ;
- temps global de stockage (attente devant les machines) [WSL95] ;
- taille des stocks, dimensionnement des lots [LSS93, Özd96, GOT93];
- nombre de changements d'outils [HW96, Özd96].

Il existe bien d'autres critères : par exemple, l'objectif peut consister à minimiser les transports (de matières premières ou de produits intermédiaires) ... Deux familles de critères peuvent être distinguées : les critères *réguliers* et les critères *non réguliers*. Un critère est dit régulier s'il est une fonction décroissante des dates de fin d'exécution des opérations. Par exemple, la durée globale de fabrication[1] d'un produit est un critère régulier.

Définition 30 (critère régulier)
Soient $E_j(x)$ et $C(x)$, respectivement la date de fin de la dernière opération du produit J_j et la valeur d'un critère C appliqué à un ordonnancement x. Considérons deux ordonnancements x et x' ; si $E_j(x) \leq E_j(x')\ \forall i = 1, 2, \cdots n$ implique $C(x) \leq C(x')$, le critère C est dit régulier.

La résolution de problèmes réels issus de l'industrie fait bien souvent intervenir plusieurs critères qu'il s'agit d'optimiser simultanément afin d'obtenir le meilleur compromis possible. Pour de tels problèmes, le fait de réduire la durée de fabrication totale n'a pas vraiment de sens si cette amélioration conduit à une détérioration globale des autres critères [BDF97], [FD98, p87].

5.1.2 Notation classique

Une notation a été adoptée afin d'identifier rapidement un problème d'ordonnancement [GOT93, ST94, JM99]. Selon cette notation, un problème d'ordonnancement s'écrit $[n/m/X, \pi_1 \cdots \pi_k/\partial]$, où n est le nombre de produits, m le nombre de machines, X la nature du problème, $\pi_1 \cdots \pi_k$ des contraintes additionnelles dépendantes du problème traité, et ∂ le critère retenu pour exprimer le coût. La nature des problèmes X est représentée par une lettre : par exemple, F pour un *flowshop*, J pour un *jobshop*. Le coût ∂ peut intégrer de nombreux critères. Lorsqu'il est exclusivement lié à la durée globale du projet, ce coût est noté $C_{\max}$.

Nous allons maintenant présenter quatre familles représentatives[2] des problèmes d'ordonnancement.

[1] La durée globale d'un produit est le temps qui s'écoule entre la date de début de fabrication d'une première opération entrant dans sa réalisation, et la date de fin de sa dernière opération.

[2] Nous ne cherchons pas ici à établir une liste exhaustive des problèmes d'ordonnancement.

5.1.3 Problème de type flowshop

C'est une production linéaire, caractérisée par une séquence d'opérations identiques pour tous les produits (voir figure 5.1). Chaque produit passe successivement sur toutes les machines [CC88, PR72, GOT93, CP96, EL99]. Pour n produits, il y a $n!$ ordonnancements possibles.

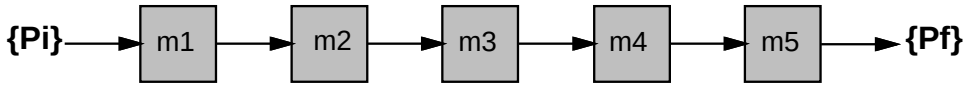


Figure 5.1: *Flowshop* – les produits empruntent le même circuit : l'ensemble des produits initiaux $\{P_i\}$ passe successivement sur toutes les machines $M_1 \cdots M_5$ pour donner un ensemble de produits finis $\{P_f\}$.

Selon la notation introduite précédemment, un problème de type *flowshop* comportant n produits à réaliser sur m machines en minimisant la durée globale du projet s'écrit $[n/m/F/C_{\max}]$.

Il existe au moins un cas où ce problème est polynomial : c'est le cas où il n'y a que deux machines $[n/2/F/C_{\max}]$. À partir de trois machines, le problème est $\mathcal{NP}$-difficile, même si les produits ne comportent pas plus de deux opérations de durée non nulle [GS78, GJ79].

Le terme *flowshops* hybrides est utilisé pour désigner des variantes de ce problème, c'est le cas lorsque les machines existent en plusieurs exemplaires [CT94, WSS91, PG95, RAE96, VDDP96], ou sont capables de réaliser simultanément plusieurs produits, ou encore lorsque les taux de rejet sont pris en compte [CS89].

Le *flowshop* est un cas particulier d'un problème plus général : le *jobshop*.

5.1.4 Problème de type jobshop

Contrairement au *flowshop*, dans un problème de type *jobshop* les produits ne passent pas systématiquement sur toutes les machines selon un ordre commun : dans l'atelier, chaque produit emprunte un « chemin[3] » qui lui est propre. Ce type de problème correspond généralement à une production par lot, notamment dans le cas d'une unité de production disposant de moyens polyvalents utilisés suivant des séquences différentes en fonction des produits (voir figure 5.2) [CC88, JM74, GOT93, CP96, EL99].

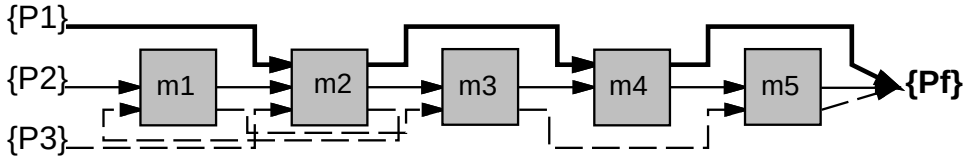


Figure 5.2: *jobshop* : les produits empruntent des chemins distincts.

Un problème de type *jobshop* comportant n produits à réaliser sur m machines en minimisant la durée globale du projet s'écrit $[n/m/J/C_{\max}]$.

La complexité du problème réside dans la gestion des contraintes d'antériorité des différentes opérations, ce qui nécessite une bonne coordination des opérations. Les contraintes d'antériorité, appelées *contraintes de précédence*, indiquent l'ordre dans lequel doivent être exécutées les opérations. Elles sont souvent définies par des inéquations de la forme $DD(O_{jk}) - DD(O_{ji}) \geq d(O_{ji})$: la $k^{\text{ème}}$ opération du produit J_j ne peut commencer avant la fin de la $i^{\text{ème}}$ opération du produit J_j. Autrement dit, entre le début de O_{jk} et le début de O_{ji} doit au moins s'écouler un temps égal à

[3] Ici un chemin est constitué d'une succession de machines.

la durée de O_{ji}. Ce type de contrainte est également appelé *contrainte potentielle* ou *contrainte de potentiel*.

Le *jobshop* est un problème $\mathcal{NP}$-dur [GS78, GJ79], très difficile à résoudre pratiquement. Il existe au moins deux cas où ce problème est polynomial : c'est le cas où il n'y a que deux machines $[n/2/J/C_{\max}]$ et le cas où il n'y a que deux produits $[2/m/J/C_{\max}]$. Le problème n'est plus polynomial dès que le nombre de machines est supérieur à deux.

Tout comme pour le *flowshop*, de très nombreuses variantes de ce problème peuvent être définies.

5.1.5 Problème de type openshop

Un problème de type *openshop* est un *jobshop* dans lequel les contraintes de précédence sont relâchées. Autrement dit, les opérations nécessaires à la réalisation de chaque produit peuvent être effectuées dans n'importe quel ordre [CC88, GOT93, CP96, EL99].

Un problème de type *openshop* comportant n produits à réaliser sur m machines en minimisant la durée globale du projet s'écrit $[n/m/O/C_{\max}]$.

Le problème de type *openshop* comportant n produits à réaliser sur deux machines en minimisant la durée globale du projet est polynomial[4]. À partir de trois machines, le problème est $\mathcal{NP}$-difficile (pour des problèmes où il n'est pas possible d'interrompre une opération en cours de réalisation) [GS78]. Tout comme pour le *flowshop* et le *jobshop*, il existe de nombreuses variantes de ce problème.

5.1.6 Problème à contraintes cumulatives

Dans les précédents problèmes, les contraintes de ressources sont de type disjonctif : deux opérations utilisant la même ressource ne pourront pas en disposer simultanément. Dans les problèmes à contraintes cumulatives, la réalisation de chaque opération nécessite plusieurs unités de certaines ressources disponibles en plusieurs exemplaires (les problèmes à contraintes de ressources disjonctives sont un cas particulier de ce type de problème où les ressources existent en un seul exemplaire).

5.2 Terminologie

Avant d'entrer dans les détails, il est nécessaire de préciser quelques termes usuels.

Définition 31 (opération)
Une opération (ou tâche) est un travail élémentaire localisé dans le temps par une date de début ou de fin et dont la réalisation nécessite un certain intervalle de temps appelé durée.

Définition 32 (produit)
Un produit, également appelé « job », est le résultat de plusieurs opérations élémentaires appliquées à une matière première ou à un produit intermédiaire. Dans le cas général, ces opérations sont partiellement ordonnées dans le temps selon un graphe de précédence appelé gamme de fabrication.

Dans le cadre de notre étude, nous introduisons le terme suivant :

Définition 33 (plan de fabrication)
Un plan de fabrication est une séquence d'opérations élémentaires (totalement ordonnées dans le temps) appliquée à une matière première ou à un produit intermédiaire. Un plan de fabrication est une gamme de fabrication linéaire.

[4] Plus précisément, il peut être résolu en $O(n)$.

Pour réaliser un produit, chaque opération nécessite l'utilisation d'une ressource pendant un certain temps :

Définition 34 (ressource)
Une ressource est un moyen technique ou humain destiné à être utilisé pour la réalisation d'une opération et disponible en quantité limitée (capacité).

Définition 35 (machine)
Dans le cadre des problèmes d'atelier, le terme machine *désigne les ressources utilisées pour mener à bien le projet global.*

Il existe deux familles de problèmes d'ordonnancement qui diffèrent sur le mode d'exécution des opérations. Pour la première famille, chaque opération est considérée comme indivisible (dans le temps). Pour la seconde, chaque opération peut être interrompue. La notion de préemption permet de distinguer ces deux familles :

Définition 36 (préemption)
L'opération de préemption consiste à interrompre une opération en cours de réalisation sur une machine, avant son achèvement, pour en démarrer une autre.

Dans la suite de ce document, nous nous intéressons uniquement aux problèmes non-préemptifs.

Les contraintes inhérentes au problème traité sont de deux types : les contraintes de potentiel et les contraintes de ressource. Les contraintes de potentiel sont des contraintes de type conjonctif liées aux relations de succession entre les opérations d'un même produit. Les contraintes de ressource sont liées à l'utilisation exclusive d'une même ressource pour la réalisation des différents produits. Ces contraintes sont de type disjonctif. En effet, lorsqu'un certain nombre d'opérations, non liées entre elles par des contraintes de précédence, doivent être réalisées à l'aide d'une même ressource, leurs périodes d'exécution sont nécessairement disjointes.

Parmi les problèmes d'ordonnancement, nous nous intéressons au *jobshop* simple. Nous introduisons en ce point la terminologie liée à ce problème et ses variantes :

Définition 37 (jobshop généralisé)
Un jobshop *est dit généralisé (ou étendu) si les produits sont constitués d'un ou plusieurs plans et si les opérations sont réalisables sur une ou plusieurs machines.*

Définition 38 (jobshop acyclique)
Une instance de jobshop *est acyclique si aucun produit n'est exécuté en plus d'un exemplaire.*

Définition 39 (jobshop simple)
Un jobshop *est dit simple si chaque produit est constitué d'un seul plan de fabrication, et si chaque opération d'un plan ne peut être effectuée que sur une seule machine. Pour un tel problème, il n'existe aucune alternative sur le choix des plans et des affectations de machines. En général, un* jobshop *simple est acyclique et la préemption n'est pas autorisée.*

Une solution du *jobshop* est appelée un ordonnancement :

Définition 40 (ordonnancement)
La « fonction ordonnancement » est une application $f : \Omega \longmapsto \mathbb{N}$ *qui associe à chaque opération* $O \in \Omega$ *une date de début d'exécution* $DD(O)$*. Le vecteur* $\vec{s} = (DD(O_{11}), DD(O_{12}), \cdots, DD(O_{JM}))$ *est appelé ordonnancement, c'est une solution du problème.*

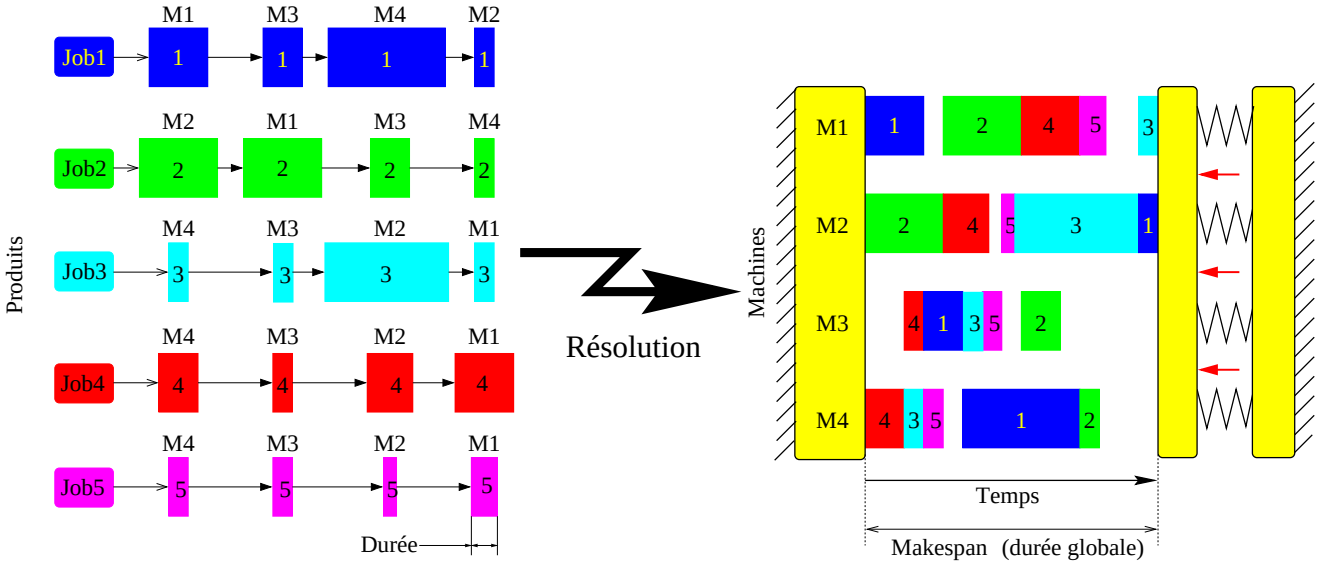


Figure 5.3: Principe de résolution du *jobshop* simple: pour ce problème, l'ordre de réalisation des opérations de chaque produit est fixé ainsi que la machine sur laquelle s'effectue chaque opération. La résolution du problème consiste à déterminer un ordre d'exécution des opérations sur chaque machine de manière à minimiser la durée globale d'exécution de l'ensemble des opérations (le makespan).

De manière générale, le coût global de production est lié au temps nécessaire à la fabrication des différents produits (voir figure 5.3). En conséquence, l'objectif consiste à réduire la durée globale de fabrication, appelée « makespan » :

Définition 41 (makespan)
Soit $E^m(x)$, la date de fin de la dernière opération réalisée sur la machine M_m selon l'ordonnancement x. Le makespan $C_{max}(x)$ de l'ordonnancement x est défini par :

$$C_{max}(x) = \max_{1 \leq m \leq M} E^m(x)$$

Le makespan est la durée nécessaire à la fabrication de tous les produits de l'instance considérée selon l'ordre de fabrication imposé par l'ordonnancement x.

Propriété 1 (makespan et critères réguliers)
Le makespan est un critère régulier (par définition).

Une opération élémentaire sur un ordonnancement consiste par exemple à avancer une opération (dans le temps) en effectuant un « décalage gauche » :

Définition 42 (décalage gauche)
Soient deux opérations O_k^m et O_j^m successivement exécutées sur la même machine M_m. Un décalage gauche consiste à avancer l'opération O_j^m en la faisant passer devant l'opération O_k^m, si O_k^m est immédiatement précédée d'une période d'inactivité de M_m supérieure ou égale à $d(O_j^m)$.

Compte tenu de la fonction coût considérée (makespan), plus un ordonnancement est « compact », meilleure est sa qualité. Les définitions suivantes permettent de préciser cette notion de « compacité ».

Définition 43 (ordonnancement sans-délai)
Un ordonnancement x de makespan $C_{max}(x)$ est un ordonnancement sans-délai ssi à tout instant t, tel que $0 \leq t \leq C_{max}(x)$ au moins une opération est en cours d'exécution sur une machine.

À chaque instant, dans un ordonnancement sans-délai, au moins une machine est active tant qu'il existe une opération en attente d'être exécutée sur l'une des machines (en respectant les contraintes de précédence).

Définition 44 (ordonnancement semi-actif)
Un ordonnancement dans lequel aucune opération ne peut être exécutée plus tôt sans altérer l'ordre d'exécution des séquences (plans) est appelé ordonnancement semi-actif.

Pour un problème d'optimisation, une solution acceptable doit être un ordonnancement semi-actif.

Définition 45 (ordonnancement actif)
Un ordonnancement tel qu'aucune opération ne puisse être exécutée plus tôt en utilisant un décalage à gauche est appelé ordonnancement actif.

Il est possible de définir un ordonnancement actif à partir d'un ordonnancement semi-actif :

*Soit un ordonnancement semi-actif x ; x est actif si, quelle que soit la machine M_m, quelles que soient deux opérations O_j^m et O_k^m telles que O_k^m soit exécutée avant O_j^m sur la machine M_m, il n'est pas possible d'effectuer un « décalage à gauche » (*i.e. *il n'est pas possible de trouver une période d'inactivité de M_m telle que O_j^m puisse être exécutée avant O_k^m).*

Ce qui revient à dire que dans un ordonnancement actif, il n'est pas possible d'avancer une opération sans en retarder une autre.

Il est possible de caractériser un sous-ensemble parmi l'ensemble des ordonnancements actifs : les « ordonnancements actifs sans-délai ».

Définition 46 (ordonnancement actif sans-délai)
Un ordonnancement actif x de makespan $C_{max}(x)$ est un ordonnancement actif sans-délai si à tout instant t, tel que $0 \leq t \leq C_{max}(x)$ aucune machine M_m n'est à l'arrêt alors qu'il existe une opération susceptible d'être effectuée sur M_m.

Propriété 2 (ordonnancements actifs et solutions optimales)
Les solutions optimales (du point de vue du makespan) sont des ordonnancements actifs et les ordonnancements actifs sont semi-actifs, mais l'inverse n'est pas toujours vrai.

Pour préciser cette notion de solutions optimales dans la propriété 2, nous introduisons la notion de *sous-ensemble dominant* : un sous-ensemble dominant est un sous-ensemble de solutions contenant au moins une solution optimale.

Propriété 3 (ordonnancements actifs et critères réguliers)
L'ensemble des ordonnancements actifs est dominant pour les critères réguliers.

La propriété 2 indique que les solutions optimales sont des ordonnancements actifs, et non uniquement des ordonnancements actifs sans-délai. Par conséquent, une méthode qui se restreint à la génération des ordonnancements actifs sans-délai peut très bien interdire la découverte d'une solution optimale.

Il est possible d'établir une « classification » des ordonnancements en fonction des propriétés précédemment énoncées (voir figure 5.4).

Soit un ordonnancement x de makespan $C_{\max}$, la marge totale définit, pour chaque opération, le retard maximum admissible sans augmentation du makespan :

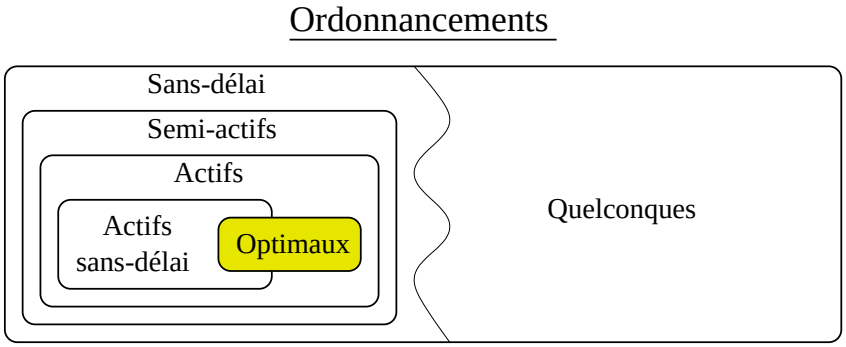


Figure 5.4: Classification des ordonnancements pour le *jobshop* simple. Les ordonnancements optimaux sont des ordonnancements actifs (éventuellement des ordonnancements actifs sans-délai). Les ordonnancements « quelconques » constituent un sous-ensemble de taille infinie. Il contient les ordonnancements jugés inacceptables (dans le cadre de la minimisation du makespan) comportant des cycles ou des temps d'arrêt de machine pouvant théoriquement s'étaler à l'infini. L'ensemble des ordonnancements sans-délai est fini (si le temps est discrétisé). Le nombre d'ordonnancements semi-actifs est inférieur à $J!^M$ pour une instance du *jobshop* composée de J jobs réalisés sur M machines (voir section 5.7).

Définition 47 (marge totale)
Pour un ordonnancement x de makespan $C_{\max}$, nous définissons tout d'abord la date de début au plus tôt, et la date de fin au plus tard d'une opération :

La date de début au plus tôt (DDT) de chaque opération O_j^m de x est obtenue en ordonnançant au plus tôt O_j^m sur la machine M_m utilisée pour son exécution (c'est-à-dire dès que M_m se libère).

La date de fin au plus tard (DFT) de chaque opération est obtenue en ordonnançant chaque opération O_j^m au plus tard selon l'ordre inverse défini par l'ordonnancement sur chaque machine et sans augmenter la valeur $C_{\max}$.

Si $d(O_j^m)$ représente la durée de l'opération O_j^m alors, la marge totale de l'opération O_j^m est égale à : $DFT(O_j^m) - \Big(DDT(O_j^m) + d(O_j^m)\Big)$.

Pour certains calculs, il peut être utile de déterminer la date de fin au plus tôt d'une opération O_j^m : $EC(O_j^m) = DDT(O_j^m) + d(O_j^m)$.

Certaines opérations ne peuvent pas être retardées sans provoquer d'augmentation de la valeur du makespan, elles sont appelées « opérations critiques » :

Définition 48 (opération critique)
Soit un ordonnancement x, une opération est dite « critique » si elle provoque l'augmentation du makespan de x lorsqu'elle est retardée (alors que l'ordre d'exécution des opérations, défini par l'ordonnancement x, ne change pas). La marge totale d'une opération critique est nulle.

Définition 49 (chemin critique)
Un chemin critique est une suite d'opérations critiques liées par des relations de précédence. La longueur d'un chemin critique est égale à la somme des durées des opérations qui le composent.

Il existe une relation entre chemin critique et makespan : la valeur du makespan est la longueur du chemin critique le plus long.

Propriété 4 (opération et chemin critiques)
Avancer une opération critique ne réduit pas la durée globale de fabrication (makespan) sauf si cette opération appartient à tous les chemins critiques.

5.3 Formulation du jobshop simple

Dans la suite de ce mémoire, nous désignons sous le terme « JSP $J\times M$ », ou « *jobshop* $J\times M$ », une instance du problème de *jobshop* simple constituée de J produits et M machines dont l'objectif consiste à minimiser le makespan[5]. Le JSP $J\times M$ peut être défini sous forme d'un couple $\langle \mathcal{M}, \mathcal{J} \rangle$:

- $\mathcal{M} = \{M_1, M_2, \ldots, M_M\}$: un ensemble de M machines ;
- $\mathcal{J} = \{J_1, J_2, \ldots, J_J\}$: un ensemble de J produits ;
  - $J_j = \{P_j\}$: chaque produit est défini par un plan de fabrication ;
  - $P_j = (\{O_{j1}, O_{j2}, \ldots, O_{jo}\}, \prec)$: chaque plan est un ensemble d'opérations sur lequel une relation de précédence est définie ;
  - $O_{jo} = \{(M_{jo}, d(O_{jo}))\}$: chaque opération est définie par un couple $\langle \text{machine}, \text{durée} \rangle$;
  - opérateur de précédence $\prec$: définit un ordre total sur les opérations d'un plan.

Nous utilisons un JSP 4×4 en guise d'exemple dans la suite de ce document :

- le produit 1 est composé des opérations 0 à 3 ;
- le produit 2 est composé des opérations 4 à 7 ;
- le produit 3 est composé des opérations 8 à 11 ;
- le produit 4 est composé des opérations 12 à 15.

N°opération	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
N°produit	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4
N°machine	3	1	4	2	4	1	2	3	1	2	3	4	1	3	2	4
Durée	54	34	61	2	9	15	89	70	38	19	28	87	95	34	7	29

Tableau 5.1: Exemple de *jobshop* – caractéristiques des opérations :
- « N°opération » est le numéro de l'opération (il apparaît une seule fois dans chaque ordonnancement) ;
- « N°produit » est le numéro du produit auquel appartient l'opération ;
- « N°machine » est le numéro de machine sur laquelle est exécutée l'opération ;
- « Durée » est la durée d'exécution de l'opération.

Les caractéristiques des opérations sont exposées dans le tableau 5.1.
La valeur minimale (optimum) du makespan pour cette instance de *jobshop* est 272.

$$(D) = \begin{pmatrix} 54 & 9 & 38 & 95 \\ 34 & 15 & 19 & 34 \\ 61 & 89 & 28 & 7 \\ 2 & 70 & 87 & 29 \end{pmatrix} \qquad (M) = \begin{pmatrix} 3 & 4 & 1 & 1 \\ 1 & 1 & 2 & 3 \\ 4 & 2 & 3 & 2 \\ 2 & 3 & 4 & 4 \end{pmatrix}$$

Figure 5.5: Exemple de *jobshop* – matrices (D) et (M) :
- $d(O_{ij})$ représente le temps nécessaire pour effectuer la $i^{\text{ème}}$ opération du produit J_j ;
- M_{ij} représente la machine utilisée pour réaliser la $i^{\text{ème}}$ opération du produit J_j.

[5]De nombreuses instances du problème défini sous cette forme sont accessibles au sein de la *OR-Library* [Bea90] via ftp à l'adresse ftp://mscmga.ms.ic.ac.uk dans le répertoire /pub.

Une instance du *jobshop* simple peut être définie par deux matrices : une matrice (D) contenant les durées et une matrice (M) contenant les machines utilisées. La figure 5.5 présente l'exemple précédent (tableau 5.1) selon cette formulation.

Le *jobshop* simple possède un certain nombre de propriétés spécifiques, liées notamment au nombre d'opérations par produit et par machine. Nous énoncerons quelques propriétés au fil de ce mémoire.

5.4 Complexité

<table>
<tr><td>Polynomial</td><td colspan="2">$\mathcal{NP}$</td></tr>
<tr><td>$M=2, \forall j\ l_j \leq 2$</td><td>(‡) $M=2, \forall j\ l_j \leq 3\ [, \exists k\ l_k = 3]$</td><td>$M=2$</td></tr>
<tr><td>$M=2, \forall o\ d(o)=1$</td><td>$M=2, \forall o\ d(o) \leq 2\ [, \exists k\ d(k)=2]$</td><td>autres cas</td></tr>
<tr><td rowspan="2"></td><td>($\star$) $M=3, \forall j\ l_j \leq 2$</td><td rowspan="2">$M \geq 3$</td></tr>
<tr><td>($\star$) $M=3, \forall o\ d(o)=1$</td></tr>
<tr><td>$J=2$</td><td colspan="2">$J \geq 3$</td></tr>
<tr><td>$C^*_{\max} \leq 3$</td><td colspan="2">$C^*_{\max} > 3$</td></tr>
</table>

Tableau 5.2: Complexité du JSP $J \times M$: l_j est le nombre d'opérations (de durée non nulle) du produit J_j, $d(o)$ la durée de l'opération o, et $C^*_{\max}$ le makespan d'un ordonnancement optimal.
(‡) le problème est $\mathcal{NP}$ même lorsque $M=2$, $J-1$ produits comportent une seule opération de durée non nulle ($l_j=1$) et un seul produit comporte trois opérations de durée non nulle ($l_k=3$) [GS78].
($\star$) ces cas sont mentionnés pour bien illustrer le fait qu'ils sont $\mathcal{NP}$.

La complexité du *jobshop* simple est fonction du nombre de produits, du nombre de machines, du nombre d'opérations par produit, et de la durée des opérations. Le tableau 5.2, inspiré de [VAL96, GS78, GJ79], présente les cas où le problème de décision associé au *jobshop* simple peut être résolu en temps polynomial et les cas où il devient $\mathcal{NP}$.

Soit une unité de production modélisée sous forme d'un *jobshop*, le problème de décision $D(R)$ associé au problème de recherche R consiste à répondre à la question suivante : soit $C \in \mathbb{N}$, existe-t-il un ordonnancement x de cette instance de *jobshop* possédant un makespan tel que $C_{\max}(x) \leq C$?

Il existe une relation entre le problème de décision $D(R)$ ci-dessus et le problème de recherche R associé au problème d'optimisation O faisant l'objet de notre étude : si le problème de décision $D(R)$ associé à R est $\mathcal{NP}$-complet, alors R est $\mathcal{NP}$-difficile.

Bien que $D(R)$ soit $\mathcal{NP}$-complet pour $M=2$ et $\forall j\ l_j > 3$, le problème O peut être résolu en temps polynomial pour $M=2$ ou $J=2$, ce qui correspond aux problèmes $[n/2/J/C_{\max}]$ ou $[2/m/J/C_{\max}]$.

5.5 Représentations des ordonnancements

Avant d'aborder les modélisations du *jobshop* simple, nous décrivons dans cette section quelques représentations usuelles des ordonnancements.

5.5.1 Matrice de séquences

Les matrices de séquences sont une représentation simple des ordonnancements :

Définition 50 (matrice de séquences)
La matrice des séquences d'opérations est définie par l'ordre des opérations sur chaque machine.

Cette représentation est incomplète car elle ne contient aucune information sur les dates de début des opérations. Pour obtenir un ordonnancement à partir de cette représentation, il suffit de définir un décodeur associé à la matrice de séquences, c'est-à-dire d'adopter une convention permettant de construire un ordonnancement réalisable à partir d'une matrice de séquences. Par exemple, ordonnancer au plus tôt chaque opération sans changer l'ordre imposé par la matrice de séquences sur chaque machine, en ordonnançant en parallèle (au plus tôt) les opérations qui ne sont pas liées par des contraintes.

Machine	Opérations			
0	8	12	5	1
1	9	6	14	3
2	0	10	7	13
3	4	11	2	15

Figure 5.6: Exemple de matrice de séquences associée à un ordonnancement d'une instance de *jobshop* 4×4.

En guise d'exemple, la figure 5.6 présente une solution potentielle du *jobshop* 4×4 décrit par le tableau 5.1 (page 57) représentée sous forme de matrice de séquences.

5.5.2 Diagramme de Gantt

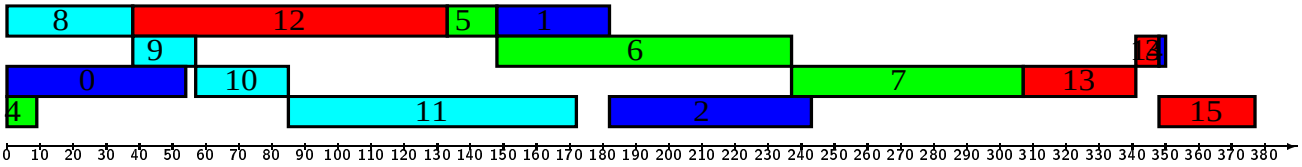


Figure 5.7: Exemple de diagramme de Gantt relatif à la matrice de séquences correspondant à la figure 5.6 : les machines sont en ordonnée, le temps en abscisse ; $C_{\max} = 377$.

Un diagramme de Gantt est une représentation dans laquelle le temps est situé en abscisse, et les produits (ou les machines) en ordonnée (voir figure 5.7).

5.6 Modélisations

Dans cette section, nous insistons plus particulièrement sur les modélisations du *jobshop* simple utilisées dans le cadre de nos travaux. Ces différentes modélisations sont à la base de méthodes de résolution présentées dans le chapitre suivant.

5.6.1 Graphe disjonctif

La modélisation du *jobshop* simple peut être effectuée sous forme de graphe disjonctif. Avant d'aborder cette modélisation, nous introduisons un certain nombre de définitions préliminaires :

Définition 51 (paire de disjonction)
Deux opérations sont en disjonction, ou forment une paire de disjonction, si ces opérations ne peuvent être exécutées simultanément. Par extension, dans une représentation sous forme de graphe, un arc est dit disjonctif s'il relie deux sommets relatifs à deux opérations ne pouvant être exécutées simultanément. Dans une telle représentation, une paire de disjonction est représentée par une arête supportant deux arcs disjonctifs orientés dans des directions opposées.

Définition 52 (clique de disjonction)
Une clique de disjonction est un ensemble de sommets tous reliés entre eux par des paires de disjonction.

Définition 53 (graphe orienté)
Un graphe orienté $\mathcal{G}$ est un couple $\langle S, A\rangle$, où S est un ensemble fini de sommets, et A un ensemble fini de paires ordonnées de sommets appelées arcs.

Définition 54 (graphe conjonctif)
Un graphe conjonctif est un graphe valué $\mathcal{G} = \langle S, C\rangle$ ayant une racine O et une antiracine A, tel qu'il existe un chemin de valeur positive entre la racine et tout autre sommet différent de la racine et l'antiracine. S est un ensemble fini de sommets ; C est un ensemble fini d'arcs.

Les arcs $(i, j) \in C$ d'un graphe conjonctif $\mathcal{G} = \langle S, C\rangle$ seront désignés sous le terme d'arcs conjonctifs dans la suite de ce mémoire.

Un graphe disjonctif est un graphe orienté ayant des paires et des cliques de disjonction [GM95]. Il peut être défini par :

Définition 55 (graphe disjonctif)
Un graphe disjonctif $\mathcal{G}$ est un couple $\langle S, C \cup D\rangle$, où S est un ensemble fini de sommets, C un ensemble fini d'arcs conjonctifs et D un ensemble fini d'arcs disjonctifs.

Définition 56 (graphe disjonctif valué)
Un graphe disjonctif valué $\mathcal{G}$ est un triplet $\langle S, P, C \cup D\rangle$, où

- *S est un ensemble fini de sommets ;*
- *C est un ensemble fini d'arcs conjonctifs ;*
- *D est un ensemble fini d'arcs disjonctifs;*
- *P est une fonction de $C \cup D$ dans $\mathbb{R}$ qui associe un poids à chaque arc.*

5.6.1.1 Application au *jobshop* : graphe potentiels-tâches

Une instance du *jobshop* simple peut être représentée par un graphe disjonctif valué [TN92, GOT93, ST94, Pen94] $\mathcal{G} = \langle S, P, C \cup D\rangle$ où

- S est un ensemble fini de sommets. Chaque sommet représente une opération. Deux opérations fictives de début et de fin (notées O_d et O_f) sont ajoutées ;
- C représente l'ordre d'exécution des opérations pour la réalisation de chaque produit ;
- D représente les différentes séquences d'opérations utilisables sur chaque machine ;
- P est une fonction de $C \cup D$ dans $\mathbb{R}$ qui associe à chaque arc un poids correspondant à la durée de l'opération associée au sommet dont il est issu. Par convention, on associe un poids nul aux arcs issus du sommet O_d.

Une paire d'arcs disjonctifs $\{(u, v), (v, u)\}$ est dite « arbitrée » si l'un des deux arcs a été ajouté à un sous-ensemble $K \subset D$ d'arcs choisis et l'autre arc a été rejeté. En choisissant (u, v), nous établissons la précédence de O_u par rapport à O_v sur leur machine commune. Lors de l'arbitrage il faut veiller à ne pas générer de circuit de longueur positive dans le graphe. Un ordonnancement réalisable est défini par un sous-ensemble $K^* \subset D$ vérifiant les deux conditions suivantes :

1. $(u, v) \in K^*$ ssi $(v, u) \in D - K^*$;
2. le graphe dirigé $\mathcal{G}(K^*) = \langle S, P, C \cup K^* \rangle$ ne comporte aucun cycle. Le makespan d'un tel ordonnancement est égal au poids du *chemin-critique* (appelé « chemin de poids maximum ») dans $\mathcal{G}(K^*)$.

Définition 57 (graphe disjonctif arbitré)
Nous utilisons le terme « graphe disjonctif arbitré » pour désigner le graphe dirigé $\mathcal{G}(K^*) = \langle S, P, C \cup K^* \rangle$ *défini ci-dessus.*

Ce graphe correspond au graphe potentiels-tâches associé à l'instance de *jobshop* simple considérée. Chaque arc (u, v), correspond à une contrainte de potentiel de la forme $DD(O_v) - DD(O_u) \geq d(O_u)$ entre les opérations O_u et O_v.

Remarque : Dans le cadre de nos travaux, nous restreignons la définition du graphe potentiels-tâches au *jobshop* simple. Toutefois, dans un cadre plus général, une modélisation sous forme de graphe potentiels-tâches permet l'intégration de contraintes additionnelles au niveau de la succession des opérations (délais, recouvrements partiels des exécutions), et des dates de début des opérations (date de début imposée par des contraintes externes au problème modélisé).

Résoudre le *jobshop* consiste à trouver le chemin-critique minimal dans $\mathcal{G}$, c'est-à-dire un chemin critique dont le poids est minimal parmi tous les sous-ensembles K^*, satisfaisant les contraintes précédemment énoncées. À partir d'un graphe disjonctif arbitré, il est possible de calculer l'ordonnancement associé et la valeur du makespan en $O(M^2)$.

Si s est une solution d'une instance *jobshop* simple représentée par un graphe disjonctif arbitré[6] $\mathcal{G}$ et K^* l'ensemble des arcs issus de l'arbitrage des paires de disjonction, alors les propriétés suivantes sont vérifiées [TCM95, GOT93] :

Propriété 5 (sens d'un arc sur un chemin critique)
Le changement de sens d'un arc $(u, v) \in K^*$ *sur un chemin critique ne peut jamais conduire à une solution non réalisable.*

Propriété 6 (sens d'un arc hors d'un chemin critique)
Si le changement de sens d'un arc $(u, v) \in K^*$*, n'appartenant pas à un chemin critique, conduit à une solution réalisable* s' *alors un chemin critique de* s' *ne peut être plus court qu'un chemin critique de* s*.*

Propriété 7 (sens d'un arc dans un graphe disjonctif arbitré)
Soit $\mathcal{G}$ *un graphe disjonctif arbitré dont le makespan est* c*. Dans tout graphe disjonctif arbitré* $\mathcal{G}'$ *de makespan* $c' < c$*, au moins un arc disjonctif appartenant à un des chemins critiques de* $\mathcal{G}$ *est inversé.*

La figure 5.8 est une représentation sous forme de graphe disjonctif du *jobshop* 4×4 présenté dans le tableau 5.1, page 57. Dans cet exemple de graphe disjonctif $\mathcal{G} = \langle S, P, C \cup D \rangle$, nous avons :

- $C = \{(O_d, 0);\ (O_d, 4);\ (O_d, 8);\ (O_d, 12);\ (0, 1);\ (1, 2);\ (2, 3);\ (4, 5);\ (5, 6);\ (6, 7);\ (8, 9);\ (9, 10);\ (10, 11);\ (12, 13);\ (13, 14);\ (14, 15);\ (3, O_f);\ (7, O_f);\ (11, O_f);\ (15, O_f)\}$
- $D = \{[(0, 7)];\ [(0, 13)];\ [(0, 10)];\ [(7, 10)];\ [(7, 13)];\ [(10, 13)];\ [(1, 5)];\ [(1, 8)];\ [(1, 12)];\ [(5, 8)];\ [(5, 12)];\ [(8, 12)];\ [(2, 4)];\ [(2, 11)];\ [(2, 15)];\ [(4, 11)];\ [(4, 15)];\ [(11, 15)];\ [(3, 6)];\ [(3, 9)];\ [(3, 14)];\ [(6, 9)];\ [(6, 14)];\ [(9, 14)]\}$ où $[(u, v)]$ représente une paire d'arcs disjonctifs $\{(u, v); (v, u)\}$

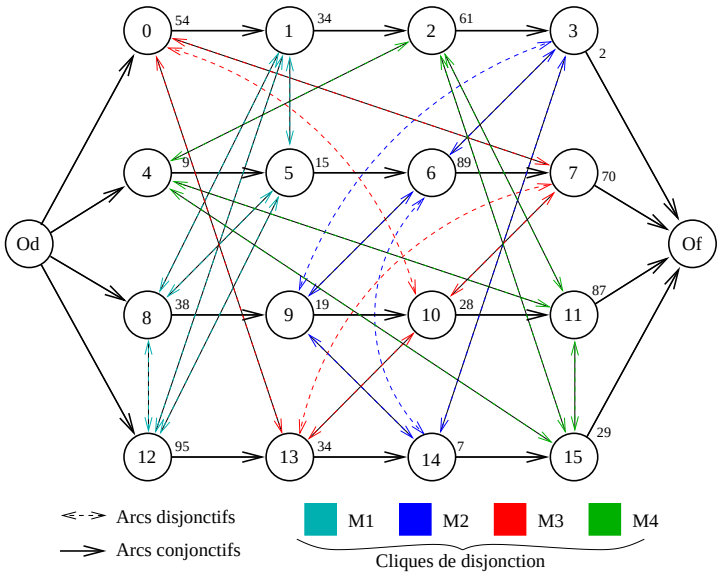


Figure 5.8: Représentation d'une instance 4×4 du *jobshop* sous forme de graphe disjonctif. Pour clarifier la représentation du graphe disjonctif, les conventions suivantes sont utilisées :
- une paire de disjonction ↑↓ est représentée par le symbole ↕;
- le poids d'un sommet correspond au poids des arcs issus de ce sommet.

Chaque ligne du graphe représente un produit, chaque nœud correspond à une opération du produit. Les nœuds sont reliés par une suite d'arcs conjonctifs représentant la suite des opérations (plan de fabrication). Les opérations sont reliées par des arcs disjonctifs représentant l'ordre d'exécution sur une machine donnée (voir figure 5.8). Sur ce graphe, le *makespan* correspond au chemin le plus long en affectant à chaque arc conjonctif la durée d'exécution de l'opération de départ.

Dans la section suivante, nous utilisons une solution potentielle du *jobshop* 4×4 décrit dans le tableau 5.1 (page 57). Représentée sous forme de matrice de séquences, et de graphe disjonctif arbitré, cette solution illustre une méthode de simplification des graphes disjonctifs arbitrés.

5.6.1.2 Simplification d'un graphe disjonctif arbitré

Il est possible de simplifier un graphe disjonctif arbitré en tenant compte des redondances de contraintes : si trois opérations 0_i, 0_j et 0_k sont liées par des arcs $(0_i,0_j)$, $(0_j,0_k)$ et $(0_i,0_k)$; alors l'arc $(0_i,0_k)$ peut être supprimé (voir figure 5.9).

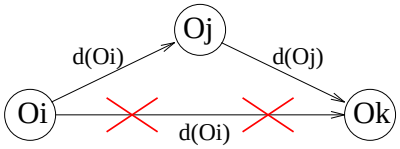


Figure 5.9: Simplification d'un graphe disjonctif arbitré : l'arc $(0_i,0_k)$ est supprimé car $d(O_i) < d(O_i)+d(O_j)$.

En fait, étant donné que le makespan correspond au chemin le plus long dans le graphe disjonctif arbitré, l'arc $(0_i,0_k)$ est inutile : les longueurs[7] associées aux arcs $(0_i,0_j)$, $(0_j,0_k)$ et $(0_i,0_k)$ sont respectivement $\mathrm{d}(0_i)$, $\mathrm{d}(0_j)$, $\mathrm{d}(O_i)$. Par conséquent :

[6] Le graphe disjonctif arbitré est supposé ne pas contenir de cycle.
[7] Rappel : par convention, on associe une longueur nulle aux arcs issus du sommet O_d.

- la longueur du chemin constitué par les arcs $(0_i,0_j)$, $(0_j,0_k)$ vaut $d(O_i) + d(O_j)$
- la longueur de l'arc $(0_i,0_k)$ vaut $d(O_i)$
- comme $d(O_j) > 0$, $d(O_i) < d(O_i) + d(O_j)$, l'arc $(0_i,0_k)$ n'appartient pas au chemin le plus long, il peut être supprimé.

5.6.1.3 Exemple de graphe disjonctif arbitré

La figure 5.10 présente une solution potentielle[8] de l'instance de JSP 4×4 définie en section 5.3 (page 57) représentée sous forme de matrice de séquences et de graphe disjonctif arbitré et simplifié.

5.6.2 Graphe potentiels-événements (PERT)

La méthode PERT[9] utilise un graphe dans lequel les sommets sont des événements (débuts d'opérations, fins d'opérations) et les arcs représentent des opérations réelles ou fictives.

À chaque opération O_{ji} sont associés deux sommets D_{ji} et F_{ji}, correspondant respectivement au début et à la fin de l'opération O_{ji}. Ces deux sommets sont reliés par un arc, allant de D_{ji} à F_{ji}, dont la longueur correspond à la durée de O_{ji}. Les contraintes de précédence sont matérialisées par des arcs fictifs de longueur nulle : pour indiquer que l'opération O_{ji} précède l'opération O_{jk} un arc fictif allant de F_{ji} à D_{jk} est ajouté au graphe.

Remarque : Le nombre de sommets est doublé par rapport à la modélisation sous forme de graphe potentiels-tâches. Certes, il est possible de simplifier ce graphe en supprimant certaines opérations fictives et en fusionnant certains événements. Toutefois, la minimisation du nombre d'arcs fictifs est un problème $\mathcal{NP}$-difficile [CC88].

Nous utilisons préférentiellement la modélisation sous forme de graphe potentiels-tâches à la modélisation sous forme de graphe potentiels-événements car à un même problème correspondent plusieurs graphes PERT plus ou moins simples. De plus, l'introduction de nouvelles contraintes pose en général des problèmes délicats et remet en question une bonne partie du graphe PERT choisi. Au contraire, la représentation par un graphe potentiels-tâches permet d'obtenir un graphe « canonique » (*i.e.* le même pour tout praticien) et d'introduire aisément des contraintes supplémentaires [CC88, page 35] et [AF90, pages 118–119].

Cette représentation offre deux avantages par rapport à la modélisation sous forme de graphe potentiels-tâches : d'une part, une opération est représentée par un seul arc ; d'autre part, si les arcs (D_{ji}, F_{ji}) sont disposés horizontalement en leur affectant une longueur proportionnelle à leur durée, il est possible d'obtenir simplement le diagramme de Gantt associé.

5.6.3 Autres modèles

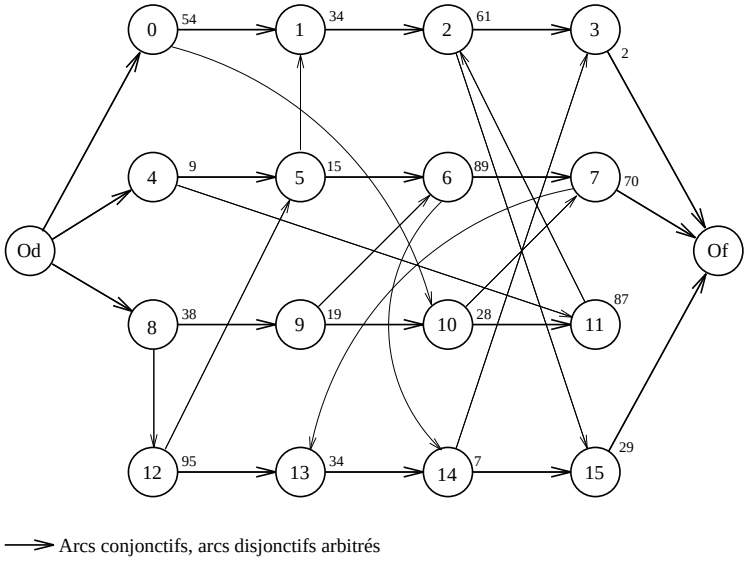
Il existe d'autres modélisations, telles que les modèles mathématiques, les réseaux de Petri temporisés, les modélisations polyédrale et algébrique [PX95, PS96, BM96, MB96a, PS96, CC88, Pen94, GOT93], ou bien encore des modélisations basées sur UML et les *design-patterns* [BFVY96, Vli98, BMMM98, WBV99, GHJV95, BRJ99, Lai00, FL00a, FL00b, MHL00]. Les réseaux de Petri

[8] Une représentation de cette solution sous forme de diagramme de Gantt a été présentée précédemment (voir figure 5.7, page 59).

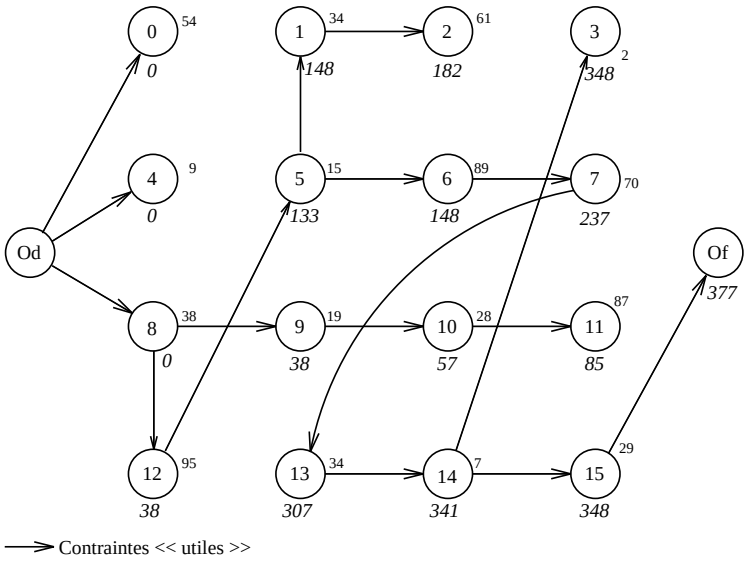
[9] Program Evaluation and Research Technique

Machine	Opérations			
0	8	12	5	1
1	9	6	14	3
2	0	10	7	13
3	4	11	2	15

(a) Matrice de séquences associée à la solution



(b) Graphe disjonctif arbitré et simplifié associé à la solution



(c) Graphe simplifié associé à la solution

Figure 5.10: Exemple de graphe disjonctif arbitré : une solution représentée sous forme de matrice de séquences en (a), est ensuite représentée sous forme de graphes disjonctifs arbitrés et simplifiés en (b) et (c) : (b) graphe disjonctif simplifié en appliquant le principe exposé en section 5.6.1.2 ;
(c) seconde simplification du graphe : seuls les arcs impliqués dans le calcul des dates de début au plus tôt des opérations sont représentés.
Pour cette solution le makespan vaut 377.

temporisés figurent parmi les premiers modèles ayant permis de représenter à la fois les contraintes de potentiel et les contraintes de ressources avec un seul formalisme graphique.

D'autres modèles, plus complets peuvent être utilisés pour modéliser le *jobshop* simple. C'est le cas par exemple du modèle RAIH[10] [Ram97, RLTP96]. Ces modèles sont dédiés à des problèmes plus complexes que le *jobshop* simple, aussi nous nous limiterons aux modèles exposés dans les sections précédentes car ils nous suffisent amplement dans le cadre de notre étude.

5.7 Nombre d'ordonnancements distincts

Avant d'aborder les méthodes de résolution du *jobshop*, nous évaluons dans cette section le nombre d'ordonnancements distincts pour une instance de *jobshop* simple de taille $J \times M$. L'objectif consiste à donner un ordre de grandeur de la taille de l'espace de recherche.

Définir un ordonnancement consiste à ordonner J opérations sur chacune des M machines. Le nombre d'ordonnancements distincts $|\mathcal{E}|$ est donc :

$$|\mathcal{E}| = J!^M$$

$|\mathcal{E}|$ est le nombre de matrices de séquences[11] distinctes pour une instance de *jobshop* donnée, mais il ne correspond pas au nombre d'ordonnancements semi-actifs distincts. En effet, ce nombre dépend du décodeur utilisé pour transformer les matrices de séquences en ordonnancements *réalisables*, c'est-à-dire ne contenant pas de cycle.

En fait, il est nécessaire de rappeler, avant de poursuivre notre raisonnement, que nous considérons des ordonnancements semi-actifs. Or pour un tel ordonnancement, représenté par une matrice de séquences, la date de début de chaque opération est imposée par les contraintes de précédence et de ressource, mais surtout par le fait d'ordonnancer au plus tôt chaque opération sans changer l'ordre imposé par la matrice de séquences sur chaque machine. De ce fait, à une matrice de séquences correspond un seul ordonnancement semi-actif *réalisable* (en l'absence de cycle). Si un cycle apparaît, il ne sera pas possible d'obtenir un ordonnancement réalisable à partir de la matrice de séquences sauf en modifiant l'ordre imposé par cette dernière.

Par conséquent, $|\mathcal{E}|$ est une borne supérieure pour le nombre d'ordonnancements semi-actifs de l'instance considérée (puisqu'il comptabilise les matrices de séquences relatives à des ordonnancements non réalisables). Il est utile de noter que $|\mathcal{E}|$ évolue plus vite en fonction de J qu'en fonction de M. Autrement dit, l'ajout d'un job a beaucoup plus d'impact sur $|\mathcal{E}|$ que l'ajout d'une machine. Dans le cadre de nos travaux nous utilisons les valeurs $|\mathcal{E}|$ pour établir un classement des instances du *jobshop* (voir section 7.2.1, page 91).

En l'absence de cycle, le mécanisme de transformation d'une matrice de séquences en ordonnancement semi-actif est bijectif. En revanche il n'en n'est rien du mécanisme de transformation d'une matrice de séquences en ordonnancement actif : transformer l'ordonnancement semi-actif, obtenu à partir de la matrice de séquences, en un ordonnancement actif nécessite généralement de permuter un certain nombre d'opérations.[12] L'ordonnancement obtenu ne correspond plus à la matrice de séquences initiale (sauf si la matrice de séquences correspond à un ordonnancement actif) : la transformation d'une matrice de séquences en ordonnancements actifs n'est pas bijective, si bien qu'il y a moins de $|\mathcal{E}|$ ordonnancement actifs pour l'instance considérée.

[10] Réseau d'Activités Incertaines Hiérarchisées

[11] voir définition 50, page 58

[12] Pour transformer une matrice de séquences en ordonnancement actif en effectuant un minimum de permutations, il est possible d'utiliser l'algorithme de Giffler et Thompson [GT69] (voir figure 6.1, page 74).

5.8 Méthodes de résolution du jobshop

Cette section présente succinctement les méthodes de résolution au *jobshop*. Nous étudions plus particulièrement différentes approches utilisées pour résoudre le *jobshop* simple afin de retenir quelques codages et plusieurs opérateurs sur lesquels centrer notre étude.

Les différentes approches existantes peuvent être réparties en plusieurs catégories [CC88, CP96, EL99, GM95, Lév94, OL96, Chu95c, Pen94, JM99]. Nous réutilisons la classification introduite par [Pen94] pour présenter les principales méthodes de résolution :

- méthodes de relaxation ;
- méthodes par décomposition ;
- méthodes exactes ;
  - programmation dynamique ;
  - procédure par séparation et évaluation (PSE) ;
  - programmation mathématique ;
- méthodes heuristiques ;
  - heuristiques constructives ;
  - heuristiques amélioratrices ;
    - ⋆ recherche tabou ;
    - ⋆ recuit simulé ;
    - ⋆ algorithmes génétiques ;
    - ⋆ colonies de fourmis ;
    - ⋆ méthodes dédiées ;
- réseaux de neurones ;
- méthodes à base de connaissances.

Dans cette classification, nous faisons abstraction des méthodes particulières utilisées pour résoudre les instances pour lesquelles le *jobshop* est polynomial. Nous préférons détailler les méthodes utilisées dans notre étude plutôt que de présenter un état de l'art complet de ces méthodes qui risquerait d'allonger considérablement ce mémoire.

Nous présentons sommairement dans le tableau 5.3 quelques applications des méthodes de résolution étudiées dans le cadre de nos travaux, en insistant plus particulièrement sur les méthodes heuristiques de types « amélioratrices ».

5.9 Conclusion

Au travers de la présentation de plusieurs modèles, ce chapitre montre qu'il est aisé de modéliser le *jobshop* simple. Dans la suite de ce document, nous avons retenu une modélisation sous forme de graphes disjonctifs car elle offre un bon compromis entre facilité d'utilisation et intégration des critères nécessaires à la suite de notre étude.

Nous avons évoqué les nombreuses applications de méthodes non dédiées à la résolution du *jobshop* afin de situer nos travaux dans ce contexte. Leur originalité ne repose pas dans l'application de telle ou telle méthode de résolution au *jobshop* car ceci a déjà été réalisé par de nombreux auteurs, mais plutôt dans les mécanismes particuliers intégrés dans ces méthodes.

Méthode de résolution	Références
algorithmes évolutifs (algorithme génétique, programmation génétique, ...)	[FB91, UBKM93, DG94, FRC94, Hus94, HMW90, TN92, NY92, NDY94, IHI96, Dav85, HLP+87, NY91, HM91, BUMK91, Bru93, FRC93, DYN93, KOY95, LGWFP97, DPT95a, GPP95, DPT96a, DPT96b, DP92, Soa94, DPT95d, PG95, Ghe95, Hus93, CPP95, BWJ90, Par92, DPT+98, DPT95b, DPT95c, PT95b, HR98]
colonies de fourmis	[CDMT94]
programmation dynamique	[CC88, CP96, EL99]
recuit simulé	[HH97, IHI96, MT96, Fle95]
recherche tabou	[HW96, DPT96a, DPT96b, EAVA97, NS96b]
programmation sous contraintes, satisfaction de contraintes	[NA94, YGMTH94, Par92, CL95a, VAL96]
procédure par séparation et évaluation	[JM74, CL95b, CL95a, BP96]
méthodes à base de connaissances, systèmes experts	[GBP96, BJ96]
systèmes multi-agent	[AR96]
autres méthodes	[Ghe92, GP97, VAL96, GPSS96, GS78, SSW93, CCP94, CPP92]
méthodes hybrides	[HH97, FRC94, KOY95, DPT95a, GPP95, CL95b, DPT96a, DPT96b, PG95, BWJ90, Par92, CL95a, DPT95c, PT95b]
état de l'art	[CC88, CP96, EL99, GOT93, TCM95, RC94, Pen94, Tai93b, Chu95c, Fan94, Ram97, JM99]

Tableau 5.3: Quelques méthodes de résolution du JSP.

Après avoir étudié la modélisation du *jobshop* simple ainsi que les méthodes de résolution non dédiées, nous allons maintenant nous intéresser aux méthodes de résolution dédiées au *jobshop*.

Chapitre 6

Méthodes de résolution dédiées au jobshop

Ce chapitre présente l'application de méthodes de résolution dédiées au *jobshop*. Il se focalise sur le *jobshop* simple.

6.1 Algorithmes approchés pour le *jobshop*

Les algorithmes approchés existent en grand nombre et sont applicables à des problèmes très variés. Dans cette section, nous présentons quelques algorithmes approchés pour le *jobshop*.

Définition 58 (algorithme approché appliqué au *jobshop*)
Un algorithme approché appliqué au jobshop *est un algorithme polynomial qui calcule – à chaque exécution – un ordonnancement réalisable tel que son makespan soit au plus ρ fois le makespan optimal.*

Un résultat négatif est connu quant-aux performances (en terme de pourcentage par rapport au coût d'une solution optimale) des algorithmes approchés :

Propriété 8 (limite des algorithmes approchés pour le jobshop)
S'il existe un algorithme approché tel que $\rho < \frac{5}{4}$ pour le jobshop*, alors $\mathcal{P} = \mathcal{NP}$*

La suite de cette section est organisée comme suit : nous introduisons tout d'abord des calculs (simples) de bornes inférieures et supérieures, nous présentons ensuite les bornes de quelques algorithmes d'approximation. Nous terminons par une présentation des algorithmes communément utilisés pour le *jobshop* simple.

6.1.1 Bornes inférieures et supérieures

Avant d'aborder les algorithmes approchés, nous présentons quelques bornes inférieures ou supérieures permettant de situer les performances des méthodes de résolution par rapport à un encadrement de la valeur optimale du makespan. Certaines de ces bornes sont utilisées pour définir les bornes des algorithmes approchés. Aussi, nous reformulons les résultats de [GPSS96, SSW93], de manière à les appliquer au *jobshop* simple.

Remarque : Les bornes définies dans cette section fournissent un encadrement *a priori* [1] des valeurs de makespan de l'ensemble des ordonnancements sans-délai. Nous ne cherchons pas en ce point à définir un encadrement de la valeur optimale du makespan au moyen de bornes précises.

Il est possible de définir une borne inférieure « triviale » pour une instance de taille $J \times M$ du *jobshop* simple :

$$C^l_{\max} = \max\{P_{\max}, \Pi_{\max}\} \quad \text{avec} \quad P_{\max} = \max_{1 \leq j \leq J} \sum_{i=1}^{M} d(O_{ji}) \quad \text{et} \quad \Pi_{\max} = \max_{1 \leq m \leq M} \sum_{j=1}^{J} d(O^m_j)$$

où O_{ji} est la i$^{\text{ème}}$ opération du produit J_j ; O^m_j est l'opération du produit J_j exécutée sur la machine M_m ; $d(O)$ est la durée de l'opération O.

En fait, $P_{\max}$ est la longueur maximale des produits de l'instance considérée si nous définissons la longueur d'un produit comme la somme des durées d'exécution de ses opérations ; $\Pi_{\max}$ est la charge maximale des machines de l'instance si nous définissons la charge d'une machine M_m comme la somme des durées d'exécution des opérations exécutées sur M_m.

Nous allons maintenant calculer une borne supérieure à partir des caractéristiques (par exemple, la durée des opérations) de l'instance. La borne utilisée est la plus grande valeur de makespan parmi l'ensemble des ordonnancements sans-délai pour l'instance de *jobshop* considérée : cette valeur correspond au cas où les opérations sont ordonnancées les unes après les autres sans recouvrement (exactement une machine active à tout instant). Il suffit de calculer la somme des durées de toutes les opérations de l'instance considérée :

$$C^u_{\max} = S_{\max} = \sum_{j=1}^{J} \sum_{i=1}^{M} d(O_{ji})$$

6.1.2 Ordonnancements sans-délai, flux moyen et makespan

Il existe des résultats concernant d'autres critères que le makespan. Par exemple, les ordonnancements sans-délai fournissent une borne pour un critère nommé *flux moyen* [GS78] : pour une instance de taille $J \times M$ du *jobshop* simple, le flux moyen, ou temps de séjour, est défini par

$$FM(x) = \frac{\sum_{j=1}^{J} E_j(x)}{J}$$

où $E_j(x)$ est la date de fin de la dernière opération du produit J_j selon l'ordonnancement x.

Si $x^* \in \mathcal{E}$ est un ordonnancement tel que $x^* = \min_{x \in \mathcal{E}}\{FM(x)\}$, le flux moyen de n'importe quel ordonnancement sans-délai x est borné par $J \times FM(x^*)$:

$$\frac{FM(x)}{FM(x^*)} \leq J$$

Cette borne est loin d'être satisfaisante, mais elle permet de situer le flux moyen des ordonnancements sans-délai par rapport au flux moyen optimal. Nous verrons dans une prochaine section que les méthodes sérielles permettent d'obtenir rapidement de meilleures bornes.

Les ordonnancements sans-délai fournissent également une borne pour le makespan [GS78] : si $x^* \in \mathcal{E}$ est un ordonnancement tel que $x^* = \min_{x \in \mathcal{E}}\{C_{\max}(x)\}$, le makespan de n'importe quel ordonnancement sans-délai x est borné par $M \times C_{\max}(x^*)$:

[1] C'est-à-dire sans utiliser de méthode de recherche (en se basant uniquement sur les caractéristiques de l'instance considérée).

$$\frac{C_{\max}(x)}{C_{\max}(x^*)} \leq M$$

Cette borne permet de situer le makespan des ordonnancements sans-délai par rapport au makespan optimal. Elle est plus satisfaisante que celle qui concerne le flux moyen étant donné que pour les instances considérées M est inférieur à J. Elle reste néanmoins insuffisante pour situer le makespan obtenu via d'autres méthodes par rapport au makespan optimal.

6.1.3 Résultats de quelques algorithmes approchés

Dans cette section, nous considérons une instance de taille $J \times M$ du *jobshop* simple, pour laquelle nous définissons la valeur suivante : u est la durée maximale des opérations de l'instance considérée.

Nous présentons uniquement les bornes obtenues par quelques algorithmes approchés sans les détailler, car seules les bornes sont utiles dans le cadre de nos travaux : nous nous intéressons aux facteurs qui interviennent dans le calcul des bornes de ces algorithmes en vue de leur utilisation lors de la définition de la difficulté d'une instance du *jobshop* simple.

Pour le cas particulier du *jobshop* acyclique constitué d'opérations de durée unitaire, un ordonnancement de makespan $O(P_{\max} + \Pi_{\max})$ existe toujours. Il peut être calculé en temps polynomial. Dans les autres cas, plusieurs bornes ont été exprimées. Nous présentons les résultats issus de différents auteurs dans la suite de cette section.

Appliqué au *jobshop* simple (acyclique, non préemptif), le premier théorème issu de [GPSS96] s'énonce comme suit :

Théorème 2 (algorithmes approchés déterministes pour le *jobshop*)
Pour le jobshop simple*, il existe des algorithmes déterministes qui engendrent des ordonnancements de makespan* $O((P_{\max} + \Pi_{\max}) \cdot \rho_i)$ $(1 \leq i \leq 3)$ *:*

(a1)
$$\rho_1 = \frac{\log(M^2)}{\log\log(M^2)} \cdot \left\lceil \frac{\log(\min\{M^2, u\})}{\log\log(M^2)} \right\rceil$$

(a2)
$$\rho_2 = \frac{\log(JM)}{\log\log(JM)} \cdot \left\lceil \frac{\log(\min\{JM, u\})}{\log\log(JM)} \right\rceil$$

(b)
$$\rho_3 = \frac{\log M}{\log\log M} \cdot \log(\min\{M^2, u\})$$

Conformément à [GPSS96], dans le théorème suivant, une « forte probabilité » représente une probabilité de $1 - \epsilon$, où ϵ est une constante positive fixée aussi petite que nous le souhaitons.

Le second théorème issu de [GPSS96] s'énonce comme suit :

Théorème 3 (algorithme approché non déterministe pour le *jobshop*)
Il existe un algorithme polynomial non déterministe pour le jobshop *qui, avec une forte probabilité, engendre un ordonnancement de makespan* $O((P_{\max} + \Pi_{\max}) \cdot \rho_4)$ *:*

$$\rho_4 = \frac{\log u}{\log\log u} \cdot \left\lceil \frac{\log(\min\{M^2, u\})}{\log\log u} \right\rceil$$

Selon les valeurs de J, M, et u, les bornes (ρ) fournies par les théorèmes précédents évoluent différemment. Il faut prendre garde aux comparaisons « directes » entre ces bornes car, d'une part il s'agit de valeurs approchées, d'autre part le théorème 3 est relatif à un algorithme non déterministe

(il ne garantit pas l'obtention d'un ordonnancement dont le makespan est inférieur à la borne calculée) contrairement au théorème 2.

Bien que les valeurs de ρ issues du théorème 2 soient des ordres de grandeur et non des valeurs exactes, il est aisé de constater que les algorithmes garantissent des résultats d'autant meilleurs que u est petit. En fait, plus la valeur de u est faible, plus on s'approche du *jobshop* préemptif (où $u = 1$), ce qui facilite considérablement l'obtention d'une solution dont le makespan est proche de l'optimum[2].

Nous avons évalué les bornes de ces algorithmes sur les instances de la bibliothèque *OR-Library* [Bea90]. En pratique, les résultats montrent que, bien que coûteux en temps d'exécution, les bornes supérieures garanties par ces algorithmes sont loin d'être satisfaisantes. Nous n'avons présenté ici que les résultats relatifs à [GPSS96]. Cependant d'autres auteurs, tels que [SSW93, FS98] obtiennent des résultats similaires non détaillés dans cette section.

En conclusion, ces algorithmes approchés ont le mérite de garantir l'obtention d'une solution de coût inférieure à une borne prédéterminée, mais en pratique on préfère bien souvent des algorithmes moins coûteux permettant d'obtenir de meilleurs résultats en moyenne, sans garantie quant-à la borne supérieure du makespan des solutions obtenues.

Dans la section suivante, nous présentons les méthodes sérielles qui correspondent tout à fait à ce type d'algorithmes peu coûteux en terme de temps de calcul et d'une bonne efficacité en pratique (en terme de coût de la solution finale).

6.1.4 Méthodes sérielles

Les méthodes sérielles sont des algorithmes à base de listes de priorités. Elles s'appuient sur des listes d'opérations préalablement ordonnées en fonction de différents critères. Ces algorithmes sont particulièrement bien adaptés à la résolution de problèmes réels, apparentés à une généralisation du problème du *jobshop* simple [JM74, Ghe92, SF97, PG95, GP97, HR98]. Ces problèmes sont souvent regroupés sous le terme *jobshop généralisé*. Lors de la résolution de ces problèmes, de nombreux paramètres peuvent être pris en compte, parmi lesquels les alternatives possibles pour le choix des machines ou des plans de fabrication, ou encore les gammes de fabrication non linéaires. Dans certains cas, la dynamique du processus industriel ainsi modélisé (pannes de machines, retards divers, maintenance, gestion des stocks) [JM74, GBP96] est également prise en compte. Dans ce type de problème, il faut réagir rapidement aux changements éventuels : ce n'est pas l'optimalité de la solution fournie qui est recherchée en priorité, mais le fait de fournir rapidement une solution qui respecte les contraintes imposées (temps, ressources, coût). Les algorithmes approchés, et plus particulièrement les méthodes sérielles sont très bien adaptées à cette problématique car elles sont capables de fournir très rapidement des solutions approchées de bonne qualité (en terme de coût).

Pour le *jobshop* simple, ce choix n'est pas aussi évident, d'autant que bien souvent le critère principal retenu est l'optimalité de la solution. Or ces algorithmes ne garantissent pas, de manière générale, de pouvoir générer la solution optimale : le fait de ne pas offrir la possibilité de différer l'ordonnancement d'une opération afin de favoriser le passage ultérieur d'une opération de priorité supérieure constitue le principal défaut des algorithmes de liste. Ceci restreint le domaine des solutions engendrées à un point tel que la solution optimale puisse ne plus appartenir à ce domaine.

Dans le paragraphe suivant nous énumérons quelques méthodes sérielles classiquement utilisées. Chaque méthode est identifiée par un acronyme qui désigne la « règle » utilisée pour sélectionner la prochaine opération parmi les opérations non encore ordonnancées :

[2]Ce qui ne signifie pas pour autant qu'il soit simple de trouver une solution optimale

FIFO (*First In First Out*) – les opérations sont placées dans une file en fonction de leur ordre d'arrivée devant les machines, la prochaine opération ordonnancée est la première dans la file ;

LIFO (*Last In First Out*) – les opérations sont placées dans une pile en fonction de leur ordre d'arrivée devant les machines, la prochaine opération ordonnancée est la première dans la pile (*i.e.* la dernière arrivée devant la machine considérée) ;

EST (*Earliest Starting Time*) – la prochaine opération ordonnancée est celle dont la date de début est antérieure à celle des autres opérations ;

LST (*Latest Starting Time*) – la prochaine opération ordonnancée est celle dont la date de début est postérieure à celle des autres opérations ;

EFT (*Earliest Finish Time*) – la prochaine opération ordonnancée est celle dont la date de fin est antérieure à celle des autres opérations ;

LFT (*Latest Finish Time*) – la prochaine opération ordonnancée est celle dont la date de fin est postérieure à celle des autres opérations ;

SPT (*Smallest Processing Time*) – la prochaine opération ordonnancée est celle dont la durée est inférieure à celle des autres opérations non encore ordonnancées ;

LPT (*Longest Processing Time*) – la prochaine opération ordonnancée est celle dont la durée est supérieure à celle des autres opérations non encore ordonnancées ;

MOPNR (*Most OPeratioNs Remaining*) ou MTR (*Most Task Remaining*) – la prochaine opération ordonnancée est la première opération non encore ordonnancée du produit comportant le plus grand nombre d'opérations non encore ordonnancées ;

MWKR (*Most WorK Remaining*) ou LRT (*Longest Remaining processing Time*) – la prochaine opération ordonnancée est la première opération non encore ordonnancée du produit dont la somme des durées des opérations non encore ordonnancées est la plus grande ;

LWKR (*Least WorK Remaining*) ou SRT *Shortest Remaining processing Time* – la prochaine opération ordonnancée est la première opération non encore ordonnancée du produit dont la somme des durées des opérations non encore ordonnancées est la plus courte ;

LRM – la prochaine opération ordonnancée est la première opération non encore ordonnancée du produit dont la somme des durées des opérations non encore ordonnancées est la plus grande sans comptabiliser la durée de la première opération ;

RANDOM – la prochaine opération est choisie aléatoirement parmi les opérations non encore ordonnancées.

Il existe de très nombreuses variantes de ces méthodes, notamment des méthodes qui tiennent compte des avances/retards des opérations [SS83, WSL95, CP96, EL99], de la dynamique du problème traité [SS83, WSL95, GP97, Ghe92, HLP+87, CP96, HR98], ou qui incorporent des critères secondaires destinés à guider les choix des opérations à ordonnancer [SS83, WSL95, CL95a, CP96].

Par exemple, la méthode de Palmer (1965) ordonnance les opérations selon un ordre décroissant établi en fonction du calcul d'un *index de pente* : $\forall j \;\; 1 \leq j \leq J \;\; P(O_j^m) = \sum_{m=1}^{M} \frac{2m-M-1}{2} d(O_j^m)$

La méthode SPT permet d'obtenir rapidement[3] un ordonnancement dont le flux moyen[4] $FM(x)$ est tel que

$$\frac{FM(x)}{FM(x^*)} \leq M$$

[3] Les méthodes telles que SPT ou MWKR ont une complexité en $O(J^2M)$.

[4] Voir section 6.1.2, page 70

Pour conclure cet exposé des méthodes sérielles, il est important de souligner que l'efficacité de ces règles est difficilement prévisible. Certes, des bornes peuvent être calculées pour certaines d'entre elles, mais d'un point de vue pratique sur une instance donnée, il est difficile de prédire le comportement d'une méthode particulière.

6.1.5 Shifting Bottleneck Procedure

Cette méthode consiste à sélectionner une machine pour laquelle une relaxation du problème initial (à une machine) est résolu de manière optimale en utilisant une méthode de séparation et évaluation progressive (PSE). Les machines sont sélectionnées à tour de rôle en choisissant la machine sur laquelle la durée globale d'exécution des opérations est la plus grande (machine *goulet*). À chaque étape, les contraintes induites par l'ordonnancement des opérations d'une machine sont propagées aux autres machines et un réordonnancement est effectué s'il permet une amélioration de la solution.

6.1.6 Discussion

Parmi les nombreuses méthodes approchées pour résoudre le *jobshop*, les méthodes sérielles sont très souvent utilisées car elles sont peu coûteuses (à la fois en terme de mémoire et de temps d'exécution). Cependant la qualité des solutions obtenues (en terme de makespan) est très variable. C'est pourquoi on les retrouve assez fréquemment dans la phase d'initialisation d'une méthode hybride. La méthode *Shifting Bottleneck Procedure* est plus performante que les méthodes sérielles mais son coût (en terme de temps d'exécution) est beaucoup plus élevé.

6.2 Génération d'ordonnancements actifs

Calculer C (appelé « coupe »), l'ensemble de toutes les premières opérations non encore ordonnancées. Calculer la date de fin au plus tôt $EC(O_j^m)$ de chaque opération $O_j^m \in C$.

① Déterminer O_{j*}^{m*} telle que $EC(O_{j*}^{m*}) = \min\{EC(O_j^m) \mid O_j^m \in C\}$.

② Constituer G l'ensemble des opérations $O_j^{m*} \in C$ telles que les exécutions de O_j^{m*} et O_{j*}^{m*} se recouvrent.

③ Choisir l'une des opérations $O_{j*}^{m*} \in G$, et ordonnancer O_{j*}^{m*} au plus tôt (*i.e.* selon $EC(O_{j*}^{m*})$).

④ Mettre à jour C et EC.

⑤ Répéter ①···④ jusqu'à ce que toutes les opérations soient ordonnancées.

Figure 6.1: L'algorithme de Giffler & Thompson calcule les dates de fin des opérations $\{DF(O_j^m)\}_{1\leq j\leq J;\ 1\leq m\leq M}$ de manière à obtenir un ordonnancement actif. Cet algorithme permet d'obtenir tous les ordonnancements actifs de l'instance considérée en énumérant tous les choix possibles à l'étape ③ S'il existe plusieurs possibilités pour déterminer O_{j*}^{m*} à l'étape ①, l'une d'entre elles est choisie aléatoirement.

Le principe de la génération d'ordonnancements actifs repose sur l'algorithme de Giffler et Thompson [GT69] (voir figure 6.1). L'ensemble des ordonnancements actifs de l'instance considérée est obtenu en explorant toutes les possibilités lorsque l'algorithme doit procéder à un choix [HR98]. Une version modifiée de cet algorithme permet d'obtenir l'ensemble des ordonnancements actifs sans-délai [HR98]. Nous rappelons en ce point que les ordonnancements optimaux ne sont pas forcément des ordonnancement actifs sans-délai (voir section 5.2, page 55).

6.3 Méthode potentiels-tâches

La méthode potentiels-tâches consiste à modéliser le problème par un graphe potentiels-tâches et à calculer le chemin le plus long dans ce graphe. Ce calcul peut être effectué de différentes manières. Nous décrivons uniquement une approche de type programmation dynamique [CP96, p.199] : les sommets du graphe potentiels-tâches sont tout d'abord numérotés de 0 à $n = J \times M$ en prenant comme sommet n l'anti-racine O_f du graphe potentiels-tâches (voir figure 5.10(b), page 64). Il n'est pas nécessaire de numéroter la racine O_d du graphe potentiels-tâches.

Les notations suivantes sont utilisées : Prec(j) est l'ensemble des sommets prédécesseurs[5] du sommet j ; a_{ij} est la longueur de l'arc allant du sommet i au sommet j.

① $l(O_d) \leftarrow 0$, $S \leftarrow \{O_d\}$
② Chercher un sommet $j \notin S$ tel que $\texttt{Prec}(j) \subset S$
③ $l(j) \leftarrow \max_{i \in \texttt{Prec}(j)} (l(i) + a_{ij})$
④ $S \leftarrow S \bigcup \{j\}$
⑤ Si $S = \{O_d, 0, 1, \cdots, n\}$
    Afficher("Plus long chemin : " $l(n)$)
    (FIN)
⑥ Sinon
    Aller en ②

Figure 6.2: Calcul du chemin le plus long dans un graphe potentiels-tâches.

La longueur du plus long chemin allant du sommet O_d au sommet j, notée $l(j)_{j \in \{O_d\} \cup \{0,1,\cdots,n\}}$, est obtenue par l'algorithme correspondant à la figure 6.2. Et l'ordonnancement obtenu correspond à un ordonnancement au plus tôt, c'est-à-dire que les opérations commencent à leur date de début au plus tôt.

À partir du même graphe potentiels-tâches, il est possible de calculer un ordonnancement au plus tard, c'est-à-dire que les opérations commencent à leur date de début au plus tard . Il suffit pour cela d'utiliser un algorithme similaire calculant, cette fois, les chemins les plus longs entre tout sommet et l'anti-racine O_f.

Remarque : À partir de l'ordonnancement au plus tôt et de l'ordonnancement au plus tard, il est possible de détecter simplement les opérations critiques : il suffit de repérer les opérations pour lesquelles la date de début au plus tôt est égale à la date de début au plus tard.

6.4 Conclusion

Les méthodes de résolution dédiées au *jobshop* sont très nombreuses, surtout lorsqu'il s'agit de résoudre un problème concret. Nous nous sommes limités à la présentation de quelques unes de ces méthodes en insistant tout particulièrement sur les méthodes sérielles en raison de leur fréquentes utilisations dans les phases d'initialisation d'autres méthodes de résolution.

Dans la suite de ce mémoire, nous laisserons les méthodes dédiées au *jobshop* de côté (sauf dans le cas où elles sont insérées au sein de méta-heuristiques hybrides), pour nous concentrer sur l'étude des méta-heuristiques.

[5] L'ensemble des sommets prédécesseurs du sommet j est l'ensemble des sommets i tels qu'il existe un arc allant de i à j

Partie III

Nos travaux

Chapitre 7

Structure du *jobshop*

Avant d'appliquer les méta-heuristiques au *jobshop*, nous étudions dans ce chapitre la structure du paysage du *jobshop*. L'objectif est de dégager des informations sur cette structure en vue de les utiliser lors de la résolution du *jobshop* au moyen de méta-heuristiques.

L'étude menée dans ce chapitre se décompose en plusieurs points. En premier lieu, nous étudions le paysage du *jobshop* engendré par différents opérateurs basés sur une principe commun (consistant à permuter deux opérations sur une machine). Cette étude vise à caractériser la « structure » du paysage du *jobshop*, plus particulièrement en ce qui concerne la présence et la répartition des plateaux et des optima locaux.

Notre démarche consiste ensuite à définir divers *critères* liés à la *difficulté* de résolution des instances du *jobshop* afin de mettre en évidence d'éventuels liens entre ces critères, la structure du paysage, et la difficulté de résolution des instances.

Les informations recueillies lors des premières phases de l'étude nous amènent à considérer différentes possibilités dans le but d'améliorer les performances des méta-heuristiques appliquées au *jobshop* (voir chapitre 8) :

- nous cherchons dans un premier temps à déceler une éventuelle « structure » du paysage, commune aux opérateurs étudiés ;
- dans un second temps, nous mettons en œuvre un ensemble de fonctions *multicritères* destinées à réduire la taille des plateaux ;
- enfin, divers mécanismes empruntés à la recherche tabou (voisinage non-constant, voisinage restreint, liste des derniers mouvements effectués) sont utilisés pour étudier le déplacement des AIRL sur les plateaux.

L'application effective de ces techniques au sein des méta-heuristiques fait l'objet du prochain chapitre.

7.1 Paysage du *jobshop*

Compte tenu de la présence de contraintes de potentiels et de ressources, le paysage du JSP est difficile à analyser de manière théorique. De ce fait, les méthodes statistiques sont très souvent utilisées pour analyser le paysage du JSP [MB96b].

7.1.1 Étude préliminaire

Nous nous appuyons sur les travaux présentés dans [MB96b] pour présenter le paysage du JSP. L'étude réalisée est basée sur un opérateur dont le principe repose sur la permutation de deux opérations consécutivement exécutées sur une même machine en se restreignant aux opérations critiques. Les auteurs étudient un ensemble de 1000 solutions générées aléatoirement ainsi que 1000 optima locaux engendrés par un AIRLD sur deux familles d'instances obtenues par construction :

- les instances faciles, pour lesquelles le makespan des solutions générées aléatoirement et des solutions issues de l'AIRLD est proche du makespan optimum ;
- les instances difficiles, pour lesquelles le makespan des solutions générées aléatoirement comme le makespan obtenu par l'AIRLD sont relativement éloignés du makespan optimum.

Les mesures effectuées montrent que la distance moyenne entre les solutions $\overline{d}$ (aléatoires ou optima locaux) est plus faible pour les instances faciles. Les valeurs de $\overline{d}$ révèlent que les optima locaux sont plus proches (en terme de distance moyenne) les uns des autres que les solutions générées aléatoirement. Cependant, contrairement aux études relatives au voyageur de commerce (TSP), les valeurs obtenues montrent que les optima locaux sont répartis dans l'espace de recherche, et non regroupés sous forme d'un « massif central ».

Pour les deux familles de problèmes, une mesure d'entropie effectuée à la fois sur l'ensemble des optima locaux ainsi que sur l'ensemble des solutions initiales permet de tirer les conclusions suivantes :

- contrairement au TSP où les optima locaux possèdent un grand nombre d'arcs en commun, les optima locaux sont très différents pour le JSP (en terme d'arcs disjonctifs arbitrés partagés par les optima locaux) ;
- pour les instances faciles le nombre d'arcs disjonctifs arbitrés partagés par les solutions (générées aléatoirement ou optima locaux) est plus élevé que pour les instances difficiles.

Ce dernier point permet de lier la notion de difficulté à la notion d'arcs disjonctifs arbitrés partagés par un ensemble de solutions. De ce fait, le nombre d'arcs partagés par un ensemble de solutions, générées aléatoirement ou issue d'un AIRL, pourrait refléter dans une certaine mesure la difficulté des instances.

La répartition des optima locaux ne permet pas d'obtenir des indications précises sur la rugosité de l'espace de recherche du JSP. Pour estimer la rugosité, une mesure d'autocorrélation est calculée sur une marche aléatoire[1] dans l'espace de recherche. Soit une marche aléatoire de longueur l, dont la séquence des coûts associés aux voisins successivement visités est notée $f_{t_{(1\leq t\leq l)}}$, le coefficient d'autocorrélation sur un intervalle $h<l$ est calculé de la manière suivante :

$$\rho(h) = \frac{\frac{1}{l-h}\sum_{t=1}^{l-h}(f_t - \overline{f})(f_{t+h} - \overline{f})}{\frac{1}{l}\sum_{t=1}^{l}(f_t - \overline{f})^2}$$

La longueur h^* jusqu'à laquelle existe une corrélation est appelée longueur de corrélation du paysage adaptatif. Elle dépend du problème traité et de l'instance considérée. À notre connaissance h^* n'a pu être déterminée de manière analytique à ce jour pour le JSP, aussi nous fixons arbitrairement la valeur de h^* conformément à [MB96b], à partir de la formule $\rho(h^*) = \frac{1}{2}$.

[1] Une marche aléatoire est obtenue à partir d'un AIRL lorsque le voisin à visiter à la prochaine itération est choisi aléatoirement (indépendamment de la fonction coût).

Les résultats présentés dans [MB96b] mettent en évidence des longueurs de corrélation beaucoup plus importantes pour les instances difficiles. Ce qui signifie que la rugosité est plus faible pour les instances difficiles que pour les instances faciles (au moins en ce qui concerne les dix instances étudiées).

Afin d'étudier la rugosité au voisinage des optima locaux[2], [MB96b] étudient la longueur de marche d'un AIRLD : le nombre de pas au cours desquels l'AIRLD améliore la solution courante, ainsi que le gain relatif au cours de chaque pas, fournissent des indications sur la variation du coût (donc sur la rugosité) au voisinage des optima locaux. Les résultats obtenus mettent en évidence des valeurs similaires pour les instances faciles et les instances difficiles.

7.1.1.1 Discussion

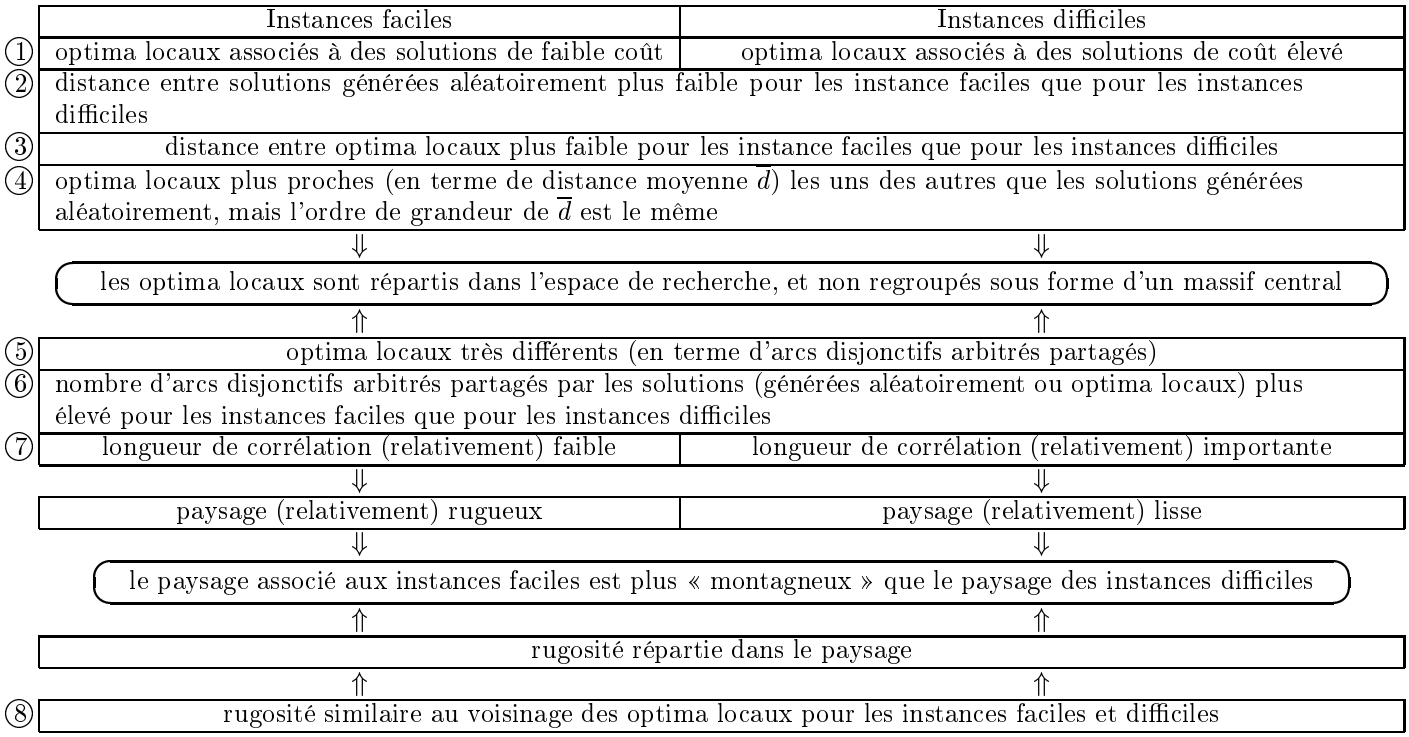


Figure 7.1: Relations entre la difficulté des instances de JSP et les paysages adaptatifs associés.

Nous synthétisons les résultats de [MB96b] dans la figure 7.1. De manière générale, pour l'opérateur considéré, l'aspect du paysage adaptatif du JSP ne facilite pas la recherche d'une solution optimale car les optima locaux sont répartis dans l'espace de recherche et les bassins d'attraction associés sont de petite taille. Pour les instances difficiles, le paysage présente une rugosité répartie dans l'espace. Une recherche locale trouve facilement une solution de bonne qualité, mais il est très difficile de trouver une solution optimale. Pour les instances faciles, le paysage présente une rugosité répartie dans l'espace à laquelle s'ajoute des monts et des vallées. Pour un tel paysage, le degré de facilité de l'instance dépend du rapport entre l'amplitude de la rugosité répartie dans l'espace (notée A_r) et l'amplitude des monts et des vallées (notée A_m). Si le ratio $\frac{A_r}{A_m}$ est très important,

[2]Cette étude permet de déterminer si la rugosité est localisée au voisinage des optima locaux, ou au contraire « répartie » sur l'ensemble du paysage.

on se ramène aux instances difficiles. Si le ratio $\frac{A_r}{A_m}$ est très faible, c'est l'aspect du paysage obtenu en considérant uniquement A_m qui va déterminer le degré de difficulté de l'instance (moyennant le fait de doter la méthode de recherche utilisée d'un mécanisme permettant de s'affranchir de A_r). Un tel mécanisme permet à la méthode de recherche d'accepter des solutions de moindre qualité dans l'espoir d'aboutir à une solution de meilleur coût ensuite. Ce type de mécanisme est implanté dans le recuit simulé et dans la recherche tabou. Nous proposons une autre technique (section 7.3, page 98) basée sur l'ajout d'informations complémentaires dans la fonction coût.

7.1.2 Étude complémentaire

① choisir une machine dans la solution initiale
② choisir deux opérations distinctes sur cette machine, considérer celle qui débute le plus tôt
③ recopier les opérations ordonnancées avant cette date dans la nouvelle solution
④ inverser les opérations sélectionnées en ② dans la liste des opérations à ordonnancer
⑤ ordonnancer toutes les opérations non ordonnancées

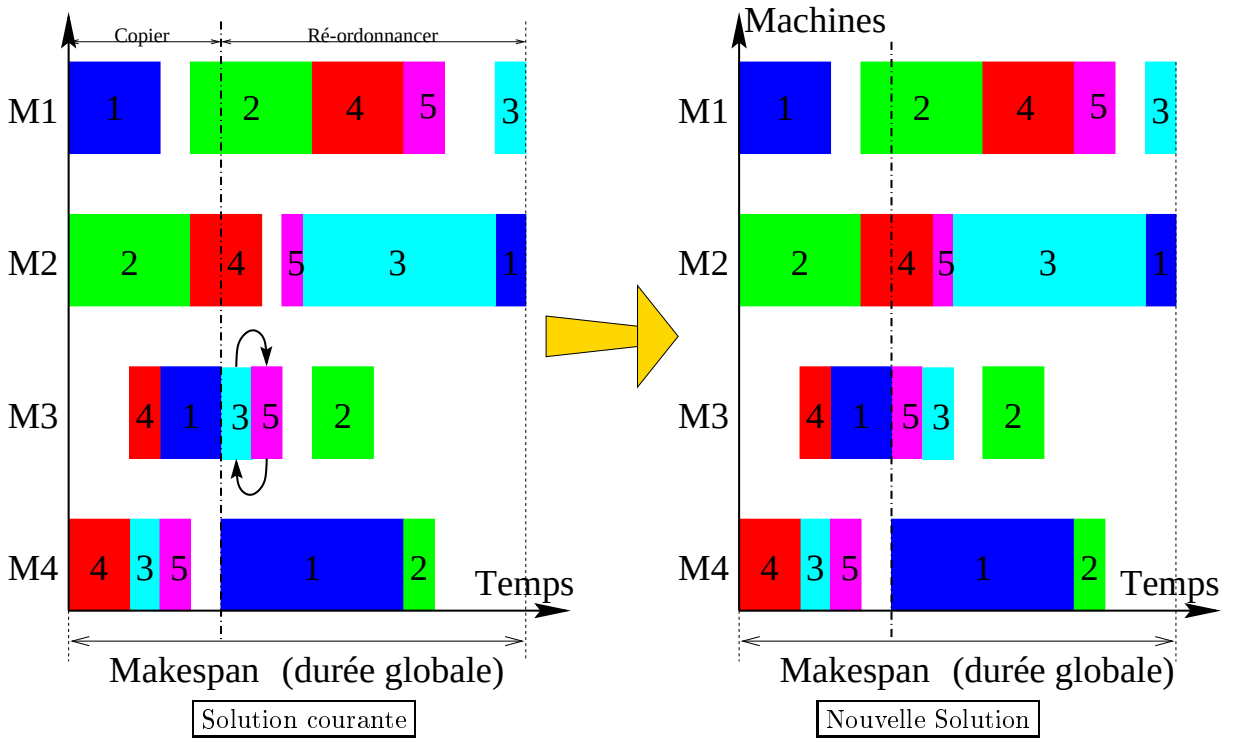


Figure 7.2: Pour cet exemple d'opérateur, la machine M_3 a été choisie à l'étape ① et les opérations relatives aux jobs 3 et 5 ont été sélectionnées à l'étape ②. L'étape ③ consiste à copier les opérations antérieures à la date de début de l'opération du job 3 sélectionnée précédemment. L'étape ④ permute les opérations relatives aux jobs 3 et 5 sur la machine M_3 en les recopiant dans la nouvelle solution. L'étape ⑤ copie les opérations qui ne figurent pas encore dans la nouvelle solution (en procédant à un réordonnancement si nécessaire).

Dans le cadre de nos travaux, nous avons utilisé trois types d'opérateurs basés sur les principes de fonctionnement suivants :

①permutation de deux opérations avec tassement[3] (voir figure 7.2) ;

②permutation de deux opérations critiques avec tassement ;

③permutation de deux opérations critiques sans tassement.

Pour les opérateurs de type ① l'opération de tassement tend à compenser le fait de considérer l'ensemble des opérations (sans se restreindre aux opérations critiques).

Le type ③ correspond exactement à l'opérateur utilisé par [MB96b].

Le fait de considérer des opérateurs incluant un mécanisme de tassement tend à rapprocher[4] artificiellement les optima locaux entre eux. La complexité des opérateurs ainsi définis rend impossible la définition d'une distance-opérateur, ce qui ne permet pas d'analyser directement le paysage obtenu. Cependant, les résultats obtenus expérimentalement à partir de ces opérateurs confirment les résultats de [MB96b] : quel que soit le type d'opérateur utilisé, le nombre de pas effectués en moyenne par un AIRLD avant arrêt de l'algorithme est extrêmement faible (*cf* table 7.8 page 110, et table 8.10 page 155). Ce résultat montre que pour un ensemble de points répartis dans le paysage, l'AIRLD est rapidement bloqué par un plateau ou par un optimum local. Ce qui confirme la présence de nombreux optima locaux répartis dans l'espace de recherche.

Afin de déterminer l'influence des plateaux sur les performances des méthodes de recherche, nous avons défini plusieurs fonctions coût dans le but de réduire la taille des plateaux sans pour autant modifier la distance-opérateur entre les optima locaux. Les résultats concernant ces fonctions seront décrits ultérieurement, mais nous pouvons en exposer brièvement le principe en terme de paysage adaptatif : ces fonctions permettent, en quelque sorte, d'ajouter du relief dans le paysage de manière à ce que là où figurait précédemment un plateau figure désormais une « *pente* » conduisant à l'optimum local le plus proche.

En assemblant les divers paramètres, nous pouvons modeler le paysage dans une certaine mesure : les modifications de la fonction coût ou la restriction de l'opérateur à la permutation des opérations critiques réduisent la taille des plateaux, tandis que l'utilisation d'une étape de tassement tend à rapprocher les optima locaux.

Il est aisé de comprendre que la réduction de la taille des plateaux tend à augmenter la « rugosité » du paysage. En revanche, l'impact de l'étape de tassement ou le cumul de plusieurs techniques sur la rugosité n'est pas simple à appréhender. Bien que les résultats présentés dans [MB96b] mettent en évidence une forte rugosité pour les instances faciles, rien n'indique *a priori* que le fait d'augmenter la rugosité au moyen de diverses techniques facilite la résolution de l'instance considérée.

Des travaux complémentaires ont été menés par de nombreux auteurs, principalement pour des problèmes réputés « plus simples[5] » que le JSP [YR97, JF95b, FRP97a, HM95, Hor96, FPRT98, MB96b, MdWS91, JF95a, REA95, Hor94, HG94, Pre99, Cha99, KH98, PRF97b, PRF$^+$99, Vas97, DW94, FRPT99, PRF97a, BPT97, Sch90, FTP97, CM91].

Nous poursuivons notre étude par une série de mesures destinées à nous guider dans le choix d'un opérateur, tout en confirmant la structure du paysage obtenu.

[3] L'opération de tassement consiste en un réordonnancement partiel des opérations si cela aboutit à une réduction du makespan. Elle peut être réalisée – par exemple – en effectuant des décalages à gauche (*cf* définition 42, page 54) tant que cela est possible (*i.e.* jusqu'à l'obtention d'un ordonnancement actif). Il est à noter que l'algorithme de Giffler et Thompson (*cf* figure 6.1, page 74) peut être utilisé pour minimiser le nombre de décalages à effectuer pour transformer un ordonnancement en un ordonnancement actif.

[4] du point de vue de la distance-opérateur définie dans le paysage considéré.

[5] au sens où un certain nombre de contraintes sont relâchées ou simplifiées

7.1.2.1 Longueur de corrélation

Nous calculons le coefficient d'autocorrélation le long de marches aléatoires pour déterminer expérimentalement la longueur de corrélation. La démarche expérimentale est basée sur les travaux de Mattfeld et Bierwirth [MB96b] : pour chaque instance, dix marches aléatoires de 100000 pas sont utilisées pour déterminer la longueur de corrélation h^* selon la méthode décrite en section 7.1.1 (page 80). Les résultats obtenus sont présentés dans le tableau 7.1.

Nous nous intéressons dans un premier temps aux instances de taille `50x15` afin de comparer nos résultats aux résultats de [MB96b][6]. La méthode utilisée par Taillard [Tai93a] pour construire les instances `ta??` de la *OR-Library* correspond à la famille des instances qualifiées de « faciles » par [MB96b]. Pour cette famille d'instances nous obtenons une longueur de corrélation moyenne de 49 contre 270 pour [MB96b].

Ce ratio, de l'ordre de 5.5 entre les deux mesures, est principalement dû à la présence d'un mécanisme de tassement dans notre opérateur. Comme expliqué précédemment (voir section 7.1.2, page 83) ce mécanisme tend à réduire la distance-opérateur entre les optima locaux. Cette réduction est confirmée par la réduction de la longueur de corrélation par rapport aux travaux de [MB96b].

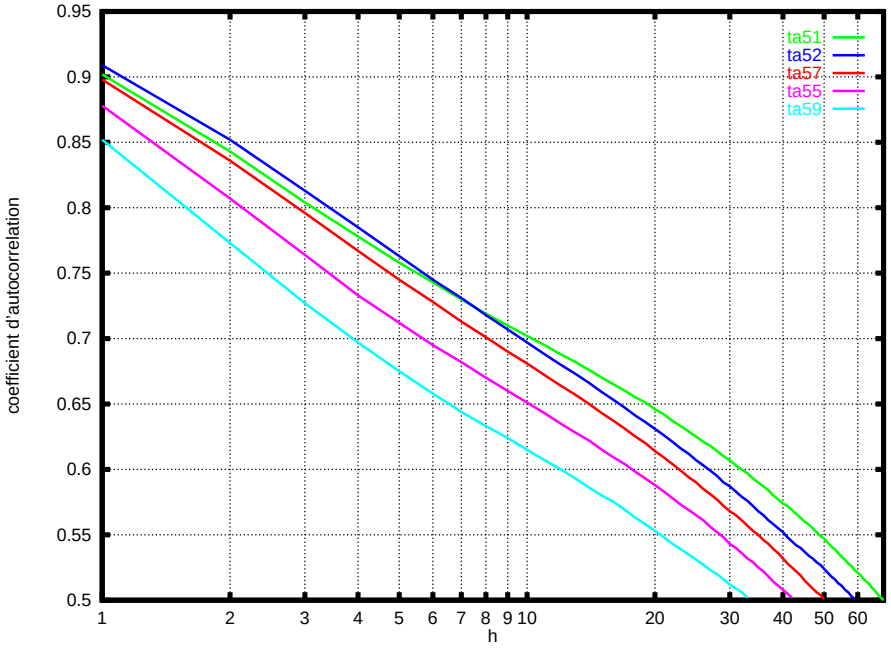


Figure 7.3: Cette figure représente l'évolution du coefficient d'autocorrélation lorsque h varie. Seules cinq instances `50x15` parmi dix sont représentées pour ne par surcharger le graphe. Les instances sont choisies de manière à représenter l'ensemble des instances (instances correspondant aux longueurs minimale, moyenne, maximale auxquelles s'ajoutent deux instances correspondant à des longueurs de corrélation intermédiaires).

Nous représentons sur la figure 7.3 l'évolution du coefficient d'autocorrélation pour cinq instances `50x15`. Les évolutions sont quasi-similaires, et le classement obtenu via les longueurs de corrélation correspond effectivement à la position relative des courbes en fin d'évolution ($h > 7$). Il est à noter cependant que les courbes relatives à `ta51` et `ta52` se coupent en début d'évolution ($h \approx 7$).

Si nous considérons l'ensemble des instances étudiées, le tableau 7.1 montre une certaine corrélation entre la taille de l'espace et la longueur de corrélation. Ceci est tout à fait compréhensible étant donné que le relief du paysage tend à se dilater en fonction de la taille de l'espace. Les valeurs

[6] [MB96b] construisent exclusivement des instances de taille `50x15`

$M\log(J!)$	h^*	dev	Inst	JxM
39	3	0.60	ft6	6x6
60	5	0.41	car7	7x7
76	5	0.63	la04	10x5
76	5	0.72	la05	10x5
76	5	0.80	la03	10x5
76	5	0.91	la01	10x5
76	6	0.91	la02	10x5
85	6	0.36	car8	8x8
88	6	0.76	car1	11x5
90	7	0.69	car2	13x4
91	7	0.58	car5	10x6
95	7	0.70	car6	8x9
100	7	0.60	car3	12x5
101	7	0.50	car4	14x4
139	6	0.83	la08	15x5
139	6	1.07	la09	15x5
139	6	1.20	la06	15x5
139	8	0.90	la07	15x5
139	8	0.91	la10	15x5
151	7	0.67	orb3	10x10
151	7	0.78	orb1	10x10
151	7	0.78	orb4	10x10
151	7	0.88	orb8	10x10
151	7	0.89	ft10	10x10
151	7	1.20	la16	10x10
151	8	0.54	orb6	10x10
151	8	0.77	orb10	10x10
151	8	0.77	orb7	10x10
151	8	0.84	la19	10x10
151	8	0.85	abz5	10x10
151	8	0.95	abz6	10x10
151	9	0.52	orb9	10x10
151	9	0.80	la20	10x10
151	9	0.88	orb2	10x10
151	9	1.03	la18	10x10
151	10	1.11	orb5	10x10
151	11	1.15	la17	10x10
212	8	1.09	la14	20x5
212	8	1.14	la11	20x5
212	9	0.75	la13	20x5
212	9	1.01	la15	20x5
212	10	0.90	ft20	20x5
212	11	0.87	la12	20x5
279	10	1.70	la23	15x10
279	11	1.02	la24	15x10
279	11	1.44	la21	15x10
279	12	1.09	la25	15x10
279	16	1.97	la22	15x10
418	10	0.64	ta06	15x15
418	11	0.65	ta10	15x15
418	11	1.28	ta08	15x15
418	12	0.95	ta09	15x15
418	12	1.58	ta01	15x15
418	12	1.58	ta07	15x15
418	12	1.63	la38	15x15
418	13	0.77	ta04	15x15
418	13	1.18	la40	15x15
418	13	1.31	ta02	15x15
418	13	1.41	ta03	15x15
418	13	1.64	ta05	15x15
418	14	1.64	la39	15x15
418	15	1.57	la37	15x15
418	17	0.96	la36	15x15
423	13	1.35	la28	20x10
423	13	1.60	la29	20x10
423	14	0.70	la27	20x10
423	15	1.20	la30	20x10
423	16	1.40	la26	20x10
635	11	0.86	ta16	20x15
635	12	1.34	ta13	20x15
635	13	1.62	ta15	20x15
635	14	1.04	abz8	20x15
635	14	1.32	ta19	20x15
635	15	1.23	ta14	20x15
635	15	1.33	ta20	20x15
635	16	0.98	abz7	20x15
635	16	1.52	ta18	20x15
635	17	1.58	ta12	20x15
635	20	0.86	ta11	20x15
635	22	1.06	ta17	20x15
635	23	2.31	abz9	20x15
747	14	1.00	la32	30x10
747	15	0.85	la33	30x10
747	16	0.89	la34	30x10
747	19	1.04	la31	30x10
747	21	2.31	la35	30x10
847	14	1.24	ta23	20x20
847	15	1.26	ta25	20x20
847	16	1.41	ta26	20x20
847	16	1.94	ta28	20x20
847	17	0.92	ta30	20x20
847	18	1.41	ta24	20x20
847	18	1.46	ta29	20x20
847	18	1.63	ta21	20x20
847	19	1.85	ta22	20x20
847	23	0.88	ta27	20x20
1120	13	1.26	ta35	30x15
1120	15	2.16	ta34	30x15
1120	16	2.23	ta40	30x15
1120	21	1.55	ta31	30x15
1120	21	2.16	ta37	30x15
1120	21	2.77	ta39	30x15
1120	22	1.75	ta33	30x15
1120	24	2.43	ta32	30x15
1120	31	1.61	ta36	30x15
1120	33	1.94	ta38	30x15
1493	18	1.27	ta50	30x20
1493	18	1.39	ta48	30x20
1493	20	1.23	ta47	30x20
1493	20	2.13	ta44	30x20
1493	20	2.20	ta49	30x20
1493	21	1.77	ta42	30x20
1493	22	1.49	ta43	30x20
1493	27	1.11	ta45	30x20
1493	28	2.11	ta46	30x20
1493	29	0.94	ta41	30x20
2227	33	3.09	ta59	50x15
2227	35	1.43	ta56	50x15
2227	42	2.38	ta55	50x15
2227	45	2.56	ta60	50x15
2227	47	1.41	ta58	50x15
2227	50	2.10	ta57	50x15
2227	53	1.99	ta54	50x15
2227	56	2.00	ta53	50x15
2227	59	1.73	ta52	50x15
2227	69	2.84	ta51	50x15

Tableau 7.1: Longueur de corrélation de quelques instances de la *OR-library*. Pour chaque instance la longueur de corrélation est obtenue en calculant un coefficient d'autocorrélation moyen sur dix marches de 100000 pas, via la formule $\rho(h) = \frac{\frac{1}{l-h}\sum_{t=1}^{l-h}(f_t-\overline{f})(f_{t+h}-\overline{f})}{\frac{1}{l}\sum_{t=1}^{l}(f_t-\overline{f})^2}$. La longueur de corrélation h^* est obtenue à partir de la formule $\rho(h^*) = \frac{1}{2}$. Les instances sont triées en fonction du logarithme de la taille de l'espace de recherche ($M\log(J!)$), puis par longueur de corrélation croissante. La colonne dev, donne la valeur de l'écart-type multiplié par 100 obtenu lors du calcul de la moyenne de $\rho(h^*)$ sur les 10 marches. Cette valeur fournit une indication sur la stabilité de la mesure de h^*.

des longueurs de corrélation présentées dans le tableau 7.1 doivent être considérées avec certaines précautions quant-à la précision des mesures : la démarche de Mattfeld et Bierwirth [MB96b] que nous avons suivi dans cette section, est basée sur dix marches de 100000 pas en considérant une *certaine* « isotropie »[7] du paysage adaptatif. Cependant, les mesures effectuées mettent en évidence de légères variations d'une marche sur l'autre. Ceci ne remet pas en cause l'hypothèse selon laquelle le paysage présente une *certaine* isotropie, mais les valeurs atteintes par l'écart-type (3.09% maximum) induisent des variations de plus ou moins une unité dans les valeurs des longueurs de corrélation.

Les remarques précédentes nous amènent à considérer la longueur de corrélation comme un indicateur de tendance, et non comme une mesure précise de la difficulté d'une instance (pour l'opérateur considéré). Par conséquent, nous poursuivons l'étude de la structure du paysage du *jobshop* en exposant la notion de rugosité du paysage.

7.1.2.2 Rugosité

La rugosité du paysage peut être définie (et mesurée) de diverses manières. Nous nous limiterons à une seule définition basée sur le calcul du coefficient d'autocorrélation d'une marche aléatoire.

Cette définition est basée sur le fait que plus un paysage est rugueux, plus l'autocorrélation d'une marche aléatoire est faible. Par conséquent nous pouvons définir la rugosité r à partir du coefficient d'autocorrélation (voir section 7.1.1, page 80) : $r = 1/\rho(1)$

7.1.3 Discussion

Deux résultats principaux ressortent de l'étude de [MB96b]. Le premier montre expérimentalement que les optima locaux sont répartis dans l'espace de recherche, et non regroupés sous forme d'un massif central. Le second met en évidence la relation entre faible rugosité du paysage et difficulté d'une instance. Ces résultats font ressortir les différences entre le paysage du JSP et le paysage d'autres problèmes tels que le TSP, le QAP ou le *flowshop* [YR97, FPRT98, FRPT99, BPT97, FTP97, Bac99]. Ils fournissent un certain nombre d'informations sur la difficulté des instances en fonction des paysages associés. Enfin, il est à noter que ces résultats se présentent plutôt sous forme de tendances que de mesures précises de la difficulté de telle ou telle instance. Ainsi par exemple, les résultats présentés dans [MB96b] mettent en évidence des longueurs de corrélation plus importantes pour les instances difficiles. D'autres indicateurs tels que l'entropie ou la longueur de marche d'un AIRL fournissent également des indications sur la difficulté des instances.

Ces mesures nécessitent un échantillonnage (statistique) de solutions dans l'espace de recherche. Il est nécessaire pour certaines d'entre elles de définir un opérateur et un AIRL.

Dans la section suivante, nous définissons des critères qui ne nécessitent pas d'échantillonnage de solutions, ni même de définir un opérateur[8].

7.2 Difficulté des instances, choix des instances

Conformément aux résultats de différents auteurs, nos résultats montrent que les instances de la *OR-Library* [Bea90] ne possèdent pas toutes la même *difficulté* face à différentes méthodes de recherche. Cette *difficulté* n'est pas directement liée à la taille de l'instance. Certaines instances

[7] Dans le cadre de cette étude, l'isotropie se traduit par une variation moyenne du coût indépendante du chemin suivit par une marche particulière.

[8] Hormis pour le critère relatif à l'AIRL-difficulté bien entendu

sont facilement résolues quelle que soit la méthode de recherche utilisée. D'autres instances sont difficiles à résoudre indépendamment de la méthode utilisée. Enfin les résultats obtenus sur un certain nombre d'instances varient en fonction de la méthode de recherche utilisée.

Afin de discerner les instances *faciles* à résoudre, nous utilisons une méthode de recherche extrêmement simple (au sens où il s'agit d'un algorithme simple) : un AIRL. L'idée sous-jacente est la suivante : si une instance est *facilement*[9] résolue par une méthode de recherche extrêmement simple, il est raisonnable de penser qu'une méthode plus complexe, capable au minimum de donner des résultats équivalents à l'AIRL, soit en mesure de résoudre *facilement* cette même instance.

Donc une première approche pour discerner les instances *faciles* des instances *difficiles* consiste à considérer qu'une instance est dite facile si un AIRL itéré trouve systématiquement[10] la solution optimale lors de chaque exécution. Une instance est difficile pour une méthode de recherche donnée si elle ne trouve pas la solution optimale.

Une autre approche peut être utilisée lorsque l'AIRL découvre rarement (*i.e.* avec une faible probabilité) une solution optimale : il s'agit de mesurer l'écart relatif entre le makespan de la meilleure solution découverte par l'AIRL et le makespan d'une solution optimale[11].

Utilisées seules, ces deux approches sont loin d'être satisfaisantes car elles sont fortement dépendantes des conditions expérimentales (*i.e.* opérateur utilisé, nombre d'expériences, limitation éventuelle du nombre d'itérations ou de la durée d'exécution de l'AIRL, ...). Aussi, de manière à obtenir un critère indépendant de la méthode de recherche utilisée, nous nous inspirons des résultats obtenus pour le problème d'affectation quadratique (QAP) [Bac99, BPT96] à partir d'une mesure appelée *flot de dominance* [VB66]. Cette mesure n'est pas directement applicable au *jobshop*, aussi nous définissons un certain nombre de critères pour tenter de déceler un critère similaire pour le *jobshop*. L'objectif consiste à définir un « indicateur » qui nous renseigne *a priori* sur la difficulté d'une instance du *jobshop*.

Dans la section 7.2.1, nous présentons différents critères à partir desquels nous tentons de définir la difficulté d'une instance. Dans une seconde étape, un AIRL sera utilisé dans le but de montrer les éventuelles relations entre les critères définis dans la section 7.2.1 et les résultats d'un AIRL.

7.2.1 Critères définis

Dans un premier temps, nous étudions les bornes supérieures et inférieures du makespan, afin de cerner l'intervalle de variation du makespan pour les différentes instances.

Nous insistons sur le fait que les bornes que nous utilisons fournissent un encadrement de l'intervalle des valeurs de makespan relatif à l'ensemble des ordonnancements sans-délai, et non un encadrement de la valeur optimale du makespan. Nous ne négligeons pas pour autant l'importance de la définition de telles bornes car l'impact de la précision de ces bornes est particulièrement sensible pour les méthodes de recherche de type *branch and bound* : les instances pour lesquelles des bornes (statiques ou dynamiques) proches de l'optimum sont rapidement obtenues sont facilement résolues par ces méthodes, en raison de l'élagage de l'arbre de recherche induit par ces bornes. À elles seules, ces bornes fournissent des indications sur la difficulté d'une instance. Nous envisageons de les intégrer ultérieurement dans le calcul de nos critères.

Nous définissons la valeur $\Delta = C^u_{\max} - C^l_{\max}$ à partir des bornes définies en section 6.1.1 (page 69). L'intervalle de variation du makespan sur l'ensemble des ordonnancements de l'instance considérée correspond à la valeur de Δ. Il dépend uniquement de l'instance de *jobshop* considérée. Afin de

[9] Autrement dit, si la méthode de recherche trouve fréquemment une solution optimale en un temps acceptable.
[10] Ou avec une forte probabilité de $1 - \epsilon$, où ϵ est une constante positive fixée aussi petite que nous le souhaitons.
[11] Cet écart est appelé *écart à l'optimum* dans la suite du manuscrit

comparer les valeurs de Δ de différentes instances entre elles, nous les normalisons par rapport à la durée moyenne des opérations de l'instance. Par conséquent, le critère Δ_r est défini par :

$$\Delta_r = \frac{\Delta}{\text{moy}} \qquad \text{avec} \quad \text{moy} = \frac{\sum_{j=1}^{J}\sum_{m=1}^{M} d(O_j^m)}{JM}$$

Δ_r donne une indication très grossière de la difficulté de l'instance considérée : si Δ_r est faible, alors tous les ordonnancements ont un makespan quasi-identique et il est facile de trouver un ordonnancement dont le makespan est proche[12] de l'optimum ; ce qui ne signifie pas qu'il soit facile de trouver un ordonnancement optimum.

De même, pour une instance du *jobshop* où les durées des opérations sont très proches, il est très facile de trouver une bonne solution initiale (puisque le makespan des solutions réalisables varie très peu). En revanche, il est difficile de trouver une solution optimale puisque le makespan ne guide pas efficacement les méthodes de recherche dans l'espace de recherche (nous faisons abstraction des cas limites où toutes les opérations ont la même durée). Afin d'étudier les variations des durées des opérations, nous avons défini le critère $\sigma_{r\%}$ en normalisant l'écart-type des durées des opérations par rapport à la moyenne des durées des opérations pour comparer les instances entre elles :

$$\sigma_{r\%} = 100 \cdot \frac{\sigma}{\text{moy}} \qquad \text{avec} \qquad \sigma = \sqrt{\frac{\sum_{j=1}^{J}\sum_{m=1}^{M}\left(d(O_j^m)-\text{moy}\right)^2}{JM}}$$

Ce critère est inspiré d'une mesure, utilisée pour résoudre le problème d'affectation quadratique (QAP), appelée « flot de dominance » [VB66, BPT96]. Contrairement à Δ_r, la valeur de $\sigma_{r\%}$ n'a de sens que si « les opérations ne sont pas trop interdépendantes ». La figure 7.4 permet d'illustrer notre propos. Les graphes disjonctifs (1) et (2) sont associés à deux instances possédant la même valeur de $\sigma_{r\%}$; la différence entre (1) et (2) réside dans les machines utilisées pour réaliser les opérations, ce qui induit une interdépendance plus ou moins forte entre les opérations : le retard d'une opération est susceptible de retarder un nombre plus ou moins important d'opérations. Pour illustrer cette différence, nous énumérons dans le tableau 7.4(b) les opérations susceptibles d'être affectées[13] par le retard de chaque opération : sans faire de supposition préalable sur l'arbitrage des paires de disjonctions, le nombre moyen d'opérations susceptibles d'être affectées par le retard d'une opération vaut respectivement 9 pour le graphe (1) et 6 pour le graphe (2), ce qui signifie que l'interdépendance entre les opérations est plus importante dans le graphe (1) que dans le graphe (2).

Afin de mettre en évidence cette interdépendance entre opérations, nous avons défini deux familles de critères :

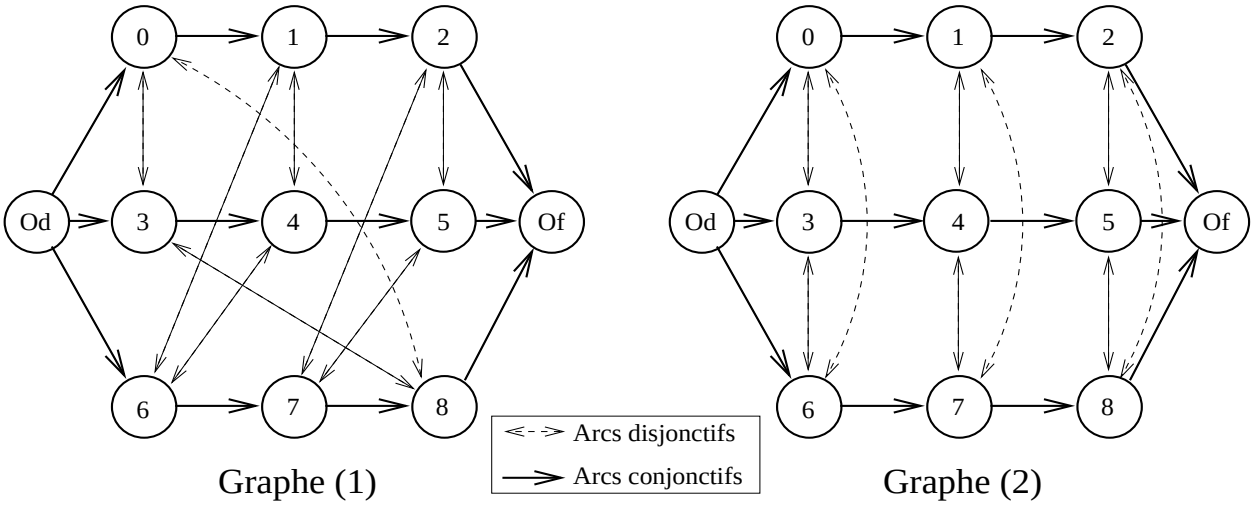
- une famille de critères est basée sur des mesures statistiques de type « écart-type » plus précises que $\sigma_{r\%}$;
- une famille de critères est basée sur les propriétés des paires de disjonctions.

Nous allons maintenant exposer les critères relatifs à la première famille. Ces critères sont conçus de manière à étudier plus précisément la variation des durées des opérations par produit ou par machine. Ils sont définis de la manière suivante :

- $\sigma_{mr\%}$ écart-type normalisé des écarts-types des durées des opérations groupées par machines ;
- $\sigma_{jr\%}$ écart-type normalisé des écarts-types des durées des opérations groupées par produits ;

[12] En pourcentage d'écart à la valeur du makespan d'une solution optimale.

[13] Il s'agit en réalité des successeurs de chaque opération, obtenus en considérant l'ensemble des arcs disjonctifs.



(a) Exemples de graphes disjonctifs

Graphe (1)		Graphe (2)	
Op	Successeurs	Op	Successeurs
0	1,2,3,4,5,6,7,8,O_f	0	1,2,3,4,5,6,7,8,O_f
1	0,2,3,4,5,6,7,8,O_f	1	2,4,5,7,8,O_f
2	0,1,3,4,5,6,7,8,O_f	2	5,8,O_f
3	0,1,2,4,5,6,7,8,O_f	3	0,1,2,4,5,6,7,8,O_f
4	0,1,2,3,5,6,7,8,O_f	4	1,2,5,7,8,O_f
5	0,1,2,3,4,6,7,8,O_f	5	2,8,O_f
6	0,1,2,3,4,5,7,8,O_f	6	0,1,2,3,4,5,7,8,O_f
7	0,1,2,3,4,5,6,8,O_f	7	1,2,4,5,8,O_f
8	0,1,2,3,4,5,6,7,O_f	8	2,5,O_f
moy	9	moy	6
min	9	min	3
max	9	max	9

(b) Successeurs potentiels des opérations

Figure 7.4: Successeurs potentiels des opérations dans un graphe disjonctif : en fonction des arbitrages des paires de disjonctions, nous construisons la liste des opérations qu'il est possible d'atteindre à partir de chaque opération.

- $\sigma_{tr‰}$ écart-type normalisé des sommes des durées des opérations groupées par machines ;
- $\sigma_{ur‰}$ écart-type normalisé des sommes des durées des opérations groupées par produits.

Plus précisément, ces critères sont calculés de la manière suivante :

$$\sigma_{mr\%} = 100 \cdot \frac{\sigma_{m=1}^{M} \sigma_{j=1}^{J} d(O_j^m)}{\text{moy}_{m=1}^{M} \sigma_{j=1}^{J} d(O_j^m)} \qquad \sigma_{jr\%} = 100 \cdot \frac{\sigma_{j=1}^{J} \sigma_{m=1}^{M} d(O_j^m)}{\text{moy}_{j=1}^{J} \sigma_{m=1}^{M} d(O_j^m)}$$

$$\sigma_{tr‰} = 1000 \cdot \frac{\sigma_{m=1}^{M} \sum_{j=1}^{J} d(O_j^m)}{J \cdot \text{moy}} \qquad \sigma_{ur‰} = 1000 \cdot \frac{\sigma_{j=1}^{J} \sum_{m=1}^{M} d(O_j^m)}{M \cdot \text{moy}}$$

En nous basant sur la représentation du *jobshop* sous forme de graphe disjonctif (voir définition 55, page 60), nous avons défini un critère nommé *nombre d'arcs disjonctifs rétrogrades* D_r dans le but d'évaluer l'interdépendance entre les opérations d'une instance donnée. Dans un graphe disjonctif, les arcs rétrogrades sont les arcs disjonctifs dirigés de droite à gauche (de l'antiracine vers la racine). Étant donné que nous voulons définir les différents critères en considérant une instance de *jobshop* et non un ordonnancement particulier[14], nous ne pouvons faire aucune hypothèse sur l'arbitrage des paires de disjonction. Donc toute paire de disjonction non représentée verticalement sur le graphe disjonctif sera comptabilisée dans D_r. Nous définissons deux critères associés à D_r : le premier critère est le nombre d'arcs non rétrogrades $\overline{D_r}$, obtenu en incrémentant $\overline{D_r}$ chaque fois qu'une paire d'arcs de disjonction verticale est rencontrée en parcourant l'ensemble des paires de disjonction du graphe disjonctif. Il existe une relation simple entre D_r et $\overline{D_r}$:

$$D_r = \frac{MJ(J-1)}{2} - \overline{D_r}$$

Le second critère intègre des informations relatives à la position dans le graphe disjonctif des paires de disjonctions, il est noté D_{wr}. Ces critères sont calculés comme suit :

$$D_r = \sum_{m=1}^{M} \sum_{i=2}^{J} \sum_{j=1}^{i-1} \mathbb{1}\left(\text{pos}(O_i^m) \neq \text{pos}(O_j^m)\right)$$
$$D_{wr} = \sum_{m=1}^{M} \sum_{i=2}^{J} \sum_{j=1}^{i-1} |\text{pos}(O_i^m) - \text{pos}(O_j^m)| \cdot (M - \text{pos}(O_i^m)) \cdot (M - \text{pos}(O_j^m))$$

$$\mathbb{1}(\text{prédicat}) = \begin{array}{l} 1 \text{ si prédicat est vrai} \\ 0 \text{ sinon} \end{array}$$

Afin de faciliter les comparaisons entre instances, nous normalisons D_r et D_{wr} :

$$\overline{D_{r\%}} = 200 \cdot \frac{\overline{D_r}}{MJ(J-1)} \qquad D_{wr\%} = 100 \cdot \frac{D_{wr}}{MJ(J-1)(M-1)(M-2)}$$

La variation des durées des opérations n'est pas le seul facteur qui influe sur la difficulté d'une instance : il faut également considérer le ratio $\frac{J}{M}$ et la durée maximale u des opérations (voir section 6.1.3). Nous utilisons un critère u_r obtenu à partir de u en normalisant par rapport à la moyenne des durées des opérations :

$$u_{r\%} = 100 \cdot \frac{u}{\text{moy}} \qquad \text{avec} \quad u = \max_{1 \leq j \leq J, 1 \leq m \leq M} d(O_j^m)$$

Du point de vue du paysage, une valeur faible de Δ_r, correspond à un paysage relativement plat (pas de grandes différences de coût sur tout le paysage). De même une faible valeur de D_r (ou D_{wr}) accompagnée d'une faible valeur de $\sigma_{r\%}$ correspond à un paysage relativement plat.

[14] Nous cherchons à définir des critères qui nous renseignent *a priori* sur la difficulté d'une instance du *jobshop*.

Une valeur élevée de D_r (ou D_{wr}) indique une forte interdépendance entre les opérations, ce qui induit une forte épistasie, généralement synonyme de difficulté pour résoudre l'instance considérée.

Plus les valeurs de D_r (ou D_{wr}) sont petites, plus l'amplitude des variations du coût dans l'espace de recherche est lié à u et σ, puisque l'interdépendance des opérations est réduite.

Nous définissons enfin deux critères supplémentaires issus des travaux présentés dans la section 6.1.3 :

- $\Pi_{\max}+P_{\max}$ donne une borne supérieure du makespan optimum des instances. Cette borne intervient dans le calcul du makespan obtenu par les algorithmes approchés (ρ-algorithme) : $C_{\max} = \rho_i \cdot (\Pi_{\max}+P_{\max})$.
- $\frac{p_0}{\Pi}\%$ représente la valeur $100 \cdot \frac{P_{\max}}{\Pi_{\max}}$. Ce critère donne une indication sur la répartition des durées des opérations en ce qui concerne la machine la plus chargée et le produit comportant la durée opératoire la plus longue. Ce critère reflète, dans une certaine mesure, la difficulté des instances : une faible valeur pour ce critère signifie que l'instance à tendance à être facilement résolue. Ce critère seul ne suffit pas car il est, entre autre, influencé par le ratio $\frac{J}{M}$. Cette influence est clairement exprimée dans le tableau A.6 (annexe A.2, page 177) où les instances de Taillard de taille supérieure ou égale à `50x15` correspondent systématiquement à une faible valeur pour ce critère.

Il est possible d'utiliser les valeurs de ρ, issues des théorèmes 2 et 3, comme indicateur de difficulté d'une instance, mais cela ne fournit que peu d'information « utilisable en pratique » sur une instance donnée : si la valeur de ρ est élevée alors l'instance est difficile (au moins pour l'algorithme approché). Par contre, si la valeur de ρ est faible, on peut simplement affirmer qu'il est facile de s'approcher très près de l'optimum, sans aucune indication sur la facilité de trouver un optimum global.

Au cours de nos expérimentations, nous avons constaté, conformément aux résultats de nombreux auteurs, que la taille des instances n'influe pas directement sur leur difficulté. Cependant, en moyenne, plus la taille de l'espace de recherche est grande, plus le temps de recherche d'un optimum à de chances d'être long. Nous donnons des ordres de grandeur relatifs au nombre de matrices de séquences (voir section 5.7, page 65) en fonction de J et M et nous établissons un classement par ordre croissant dans le tableau 7.2. À ce propos, lorsqu'il s'agit d'établir un classement des instances de la *OR-Library*, nous utilisons le critère $\log(J!^M) = M\log(J!)$. Ce calcul permet d'obtenir un critère de classement qui demeure inférieur à 10^4 pour des instances d'une taille maximale 100×20 et s'avère plus facile à manipuler que les valeurs atteintes par $J!^M$.

Le tableau 7.3 montre que J et M n'ont pas des rôles symétriques vis-à-vis du nombre de matrices de séquences : une augmentation de J a beaucoup plus d'impact sur $|\mathcal{E}|$ qu'une augmentation de M. La même dissymétrie apparaît dans le calcul du nombre d'arcs disjonctifs ($MJ(J-1)$). Cependant, ainsi que l'affirme la conjecture de Taillard [Tai93b, p.58], une instance telle que $J \gg M$ est plus facile à résoudre en général qu'une instance telle que $J \sim M$.

Conjecture de Taillard
Les instances telles que $J \approx M$ sont plus difficiles à résoudre que celles pour lesquelles $J \gg M$; la frontière entre les instances « faciles » et les instances « difficiles » se situe aux environs de $J = 5M$.

Les critères ont été évalués sur les instances de la *OR-Library*, le détail des résultats obtenus sont rassemblés en annexe A.2 dans le tableau A.6 (page 180). Ces résultats ne permettent pas d'établir clairement un classement des 119 instances étudiées en fonction de leur difficulté. Par contre, les résultats présentés confirment la particularité des instances de Taillard (notées ta<*numéro*>) : les

J	M	$\frac{J}{M}$	$MJ(J-1)$	$\lvert\mathcal{E}\rvert = J!^M$	$M\log(J!)$
4	4	1.000	48	331776	13
6	6	1.000	180	$1,393 \cdot 10^{17}$	39
7	7	1.000	294	$8,261 \cdot 10^{25}$	60
10	5	2.000	450	$6,292 \cdot 10^{32}$	76
8	8	1.000	448	$6,985 \cdot 10^{36}$	85
11	5	2.200	550	$1,013 \cdot 10^{38}$	88
13	4	3.250	624	$1,504 \cdot 10^{39}$	90
10	6	1.667	540	$2,283 \cdot 10^{39}$	91
8	9	0.889	504	$2,816 \cdot 10^{41}$	95
12	5	2.400	660	$2,522 \cdot 10^{43}$	100
14	4	3.500	728	$5,776 \cdot 10^{43}$	101
9	8	1.125	576	$3,007 \cdot 10^{44}$	102
9	9	1.000	648	$1,091 \cdot 10^{50}$	115
15	5	3.000	1050	$3,824 \cdot 10^{60}$	139
10	10	1.000	900	$3,959 \cdot 10^{65}$	151
20	5	4.000	1900	$8,524 \cdot 10^{91}$	212
15	10	1.500	2100	$1,462 \cdot 10^{121}$	279
15	15	1.000	3150	$5,591 \cdot 10^{181}$	418
20	10	2.000	3800	$7,265 \cdot 10^{183}$	423
20	15	1.333	5700	$6,193 \cdot 10^{275}$	635
30	10	3.000	8700	$1.724 \cdot 10^{324}$	747
20	20	1.000	7600	$5.278 \cdot 10^{367}$	847
30	15	2.000	13050	$2.264 \cdot 10^{486}$	1120
30	20	1.500	17400	$2.973 \cdot 10^{648}$	1493
50	15	3.333	36750	$1.762 \cdot 10^{967}$	2227
50	20	2.500	49000	$4.587 \cdot 10^{1289}$	2970
100	20	5.000	198000	$2.512 \cdot 10^{3159}$	7275

Tableau 7.2: Classement des instances selon le nombre de matrices de séquences (log représente le logarithme népérien).
Les couples (JxM) 8x8, 8x9, 9x8 et 9x9 permettent d'illustrer la dissymétrie entre J et M

J	M	$\lvert\mathcal{E}\rvert = J!^M$
4	4	331776
4	5	7962624
5	4	207360000

Tableau 7.3: Mise en évidence de la dissymétrie entre J et M.

durées des opérations sont générées aléatoirement selon une répartition uniforme entre 1 et 100 pour toutes les instances, ce qui se traduit par une durée moyenne des opérations proche de 50 et une valeur élevée de $\sigma_{r\%}$ (supérieure à 50%) pour toutes ces instances. Or ces instances ne sont pas homogènes du point de vue de la difficulté de résolution. Cette observation permet d'affirmer que des mesures directement inspirées du « flot de dominance » (c'est-à-dire $\sigma_{r\%}$) ne suffisent pas pour évaluer la difficulté des instances.

En résumé, les résultat obtenus montrent qu'il n'existe pas un critère unique permettant de déterminer la difficulté *a priori* d'une instance de *jobshop*.

7.2.2 Utilisation d'un AIRL

De manière à compléter les résultats obtenus par différents critères, un algorithme itératif de recherche locale peut être utilisé pour déterminer statistiquement la difficulté d'une instance du *jobshop* :

Définition 59 (AIRL-difficulté du JSP)
Soient une instance I de jobshop *simple, un ensemble Λ de N solutions distinctes de I générées aléatoirement, et un AIRL $\mathcal{A}$. L'AIRL-difficulté de l'instance I de* jobshop*, notée AIRL-difficulté$(\mathcal{A}, I, \Lambda)$ est le nombre moyen d'opérations que l'AIRL permute avant d'obtenir une solution optimale à partir des solutions de Λ. Par convention AIRL-difficulté$(\mathcal{A}, I, \Lambda) = \infty$ si l'AIRL n'engendre pas de solution optimale.*

Pour obtenir l'AIRL-difficulté$(\mathcal{A}, I, \Lambda)$, l'AIRL est lancé une seule fois à partir de chacune des N solutions initiales de Λ. L'AIRL-difficulté dépend de l'opérateur ainsi que de l'AIRL utilisé, puisqu'il s'agit du nombre de permutations effectuées par cet algorithme et non du nombre *minimum d* de permutations nécessaires pour transformer une solution de Λ en solution optimale.

D'autres mesures pourraient être définies, comme par exemple le nombre de permutations nécessaires – en moyenne – le long des marches avant d'améliorer le makespan. Cette mesure serait liée à l'opérateur (plus particulièrement au paysage qu'il engendre), à la taille des plateaux, mais aussi à l'AIRL utilisé en lui-même notamment au travers du choix du voisin exploré. Un certain nombre de mesures complémentaires ont été ainsi effectuées le long des marches de l'AIRL lors d'une étude préliminaire. Toutefois, nous restreignons volontairement notre étude aux mesures directement applicables dans le cadre de la difficulté des instances du *jobshop*.

Le détail des résultats obtenus par l'AIRLNDM en utilisant f_3[‡] sur les instances de la *OR-Library* figure en annexe A.1 dans le tableau A.3 (page 172). Ces résultats permettent d'établir un classement des 119 instances étudiées en fonction de sa performance moyenne p :

- $p < 5\%$ de l'optimum pour 15 instances (la06, la11, la14, la12, la10, la05, la09, la13, la08, la07, la33, la31, la15, la01, la32) ;
- $5\% \leq p < 10\%$ de l'optimum pour 15 instances (ft6, la35, la34, la04, la17, abz6, la16, la23, la20, abz5, la03, la18, la02, la30, ta35) ;
- $10\% \leq p < 15\%$ de l'optimum pour 20 instances (ta71, la19, orb9, car5, car7, car4, orb2, car8, orb4, la26, orb7, la36, ft20, la28, la25, la21, ft10, ta06, la27, la24) ;
- $15\% \leq p < 20\%$ de l'optimum pour 30 instances (car1, car2, ta07, orb5, la37, ta02, car3, la22, orb1, ta10, la39, orb8, ta01, ta14, ta03, la40, orb6, car6, ta05, ta08, abz7, la29, ta09, ta04, orb10, ta39, ta61, ta17, la38, ta51) ;

[‡] f_3 est une fonction coût qui combine le makespan $C_{\max}$, le critère H_2 et le nombre d'opérations critiques C_{op} : $f_3(x) = k_2 \times k_1 \times C_{\max}(x) + k_2 \times H_2(x) + C_{\text{op}}(x)$ (voir la section 7.3.2.4, page 107).

– $p \geq 20\%$ de l'optimum pour les 39 instances restantes.

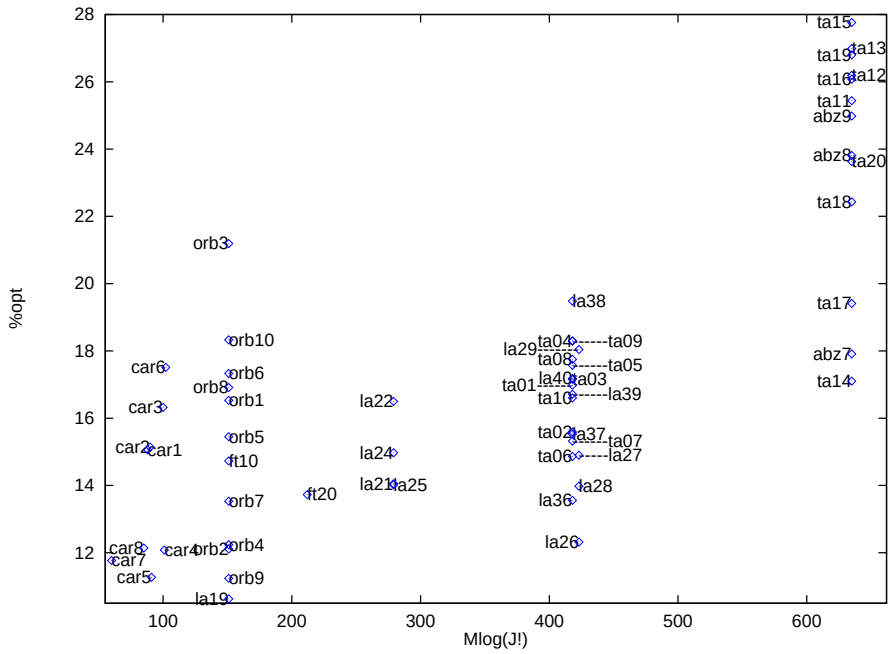


Figure 7.5: Classement des instances selon l'AIRL-difficulté : les instances sont groupées sur des lignes verticales en fonction de la taille des instances ; sur chaque ligne, l'instance de plus grande ordonnée est plus difficile (pour l'AIRLNDM(f_3)) que les autres instances de même taille. Par conséquent, car6, orb3, ft20, la22, la38, la29, ta15, ta30, ta40, ta50 sont les instances les plus difficiles pour chaque taille d'instances (au sens de l'AIRLNDM(f_3)). Les résultats détaillés figurent dans le tableau A.3, page 172.

Cette classification est loin d'être suffisante. D'une part, elle ne met pas en balance la taille de l'instance et la performance de l'AIRLND. D'autre part, étant donné la simplicité de l'AIRLND, ce dernier a tendance à donner de faibles performances pour les instances de grandes tailles (à quelques exceptions près, $p \leq 20\%$ pour les instances telles que $M \log(J!) \geq 635$). C'est pourquoi nous nous limitons aux instances telles que $M \log(J!) \leq 635$ sur la figure 7.5. Pour des raisons similaires, nous avons éliminé les instances telles que $p < 10\%$ de la figure 7.5 pour nous concentrer sur les instances de difficulté moyenne (au sens de la performance moyenne) pour l'AIRLND.

Si l'on compare la taille des instances en fonction des performances de l'AIRLNDM(f_3), nous obtenons les résultats suivants[15] :

– $p < 5\%$ de l'optimum pour 15 instances dont la taille varie entre 10x5 (76) et 30x10 (747) ;
– $5\% \leq p < 10\%$ de l'optimum pour 15 instances dont la taille varie entre 6x6 (39) et 30x15 (1120) ;
– $10\% \leq p < 15\%$ de l'optimum pour 20 instances dont la taille varie entre 7x7 (60) et 100x20 (7275) ;
– $15\% \leq p < 20\%$ de l'optimum pour 30 instances dont la taille varie entre 11x5 (88) et 50x20 (2970) ;
– $20\% \leq p < 25\%$ de l'optimum pour 15 instances dont la taille varie entre 10x10 (151) et 30x20 (1493) ;

[15] Les valeurs entre parenthèses correspondent à l'ordre de grandeur des instances : $M \log(J!)$.

- $25\% \leq p < 30\%$ de l'optimum pour 17 instances dont la taille varie entre 20x15 (635) et 30x20 (1493) ;
- $p \geq 30\%$ de l'optimum pour les 7 instances restantes dont la taille varie entre 20x20 (847) et 30x20 (1493).

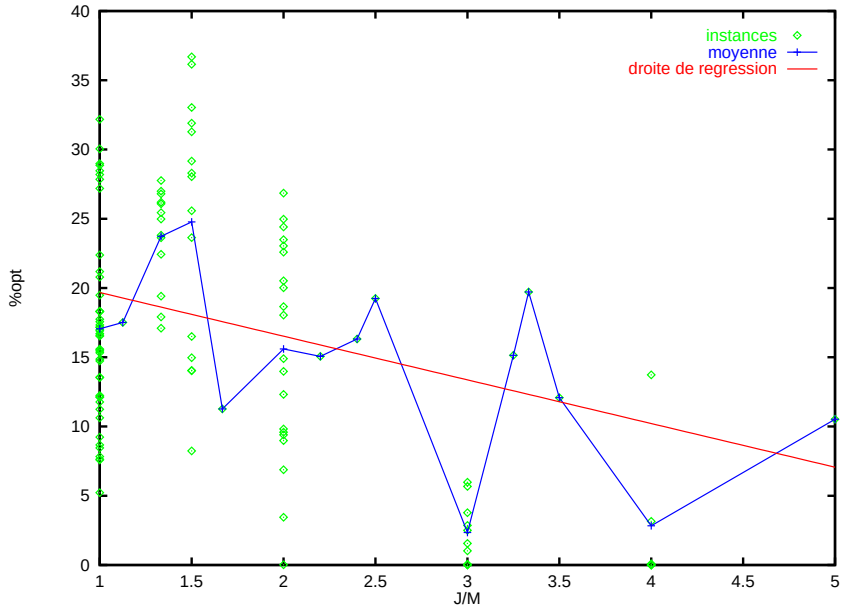


Figure 7.6: Classement des instances selon le ratio $\frac{J}{M}$: les instances sont groupées sur des lignes verticales en fonction du ratio $\frac{J}{M}$. Sur chaque ligne, la moyenne est calculée de manière à tracer l'évolution de la moyenne en fonction du ratio $\frac{J}{M}$.

Les résultats montrent qu'il n'y a pas de relation directe entre la taille d'une instance et sa difficulté, même si l'on décèle une tendance moyenne principalement due à l'inefficacité de l'algorithme utilisé (AIRL) pour les instances de grandes tailles.

De manière à pousser plus loin nos investigations, nous avons vérifié si la conjecture de Taillard (voir section 7.2.1, page 91) s'applique aux instances étudiées bien que le ratio $\frac{J}{M}$ soit inférieur à 5 : la courbe présentée sur la figure 7.6 montre une légère décroissance globale de la moyenne de l'écart à l'optimum (noté $\%_{\mathsf{opt}}$ en ordonnée) lorsque le ratio $\frac{J}{M}$ décroît. Cependant il est difficile de déceler cette tendance compte tenu des fortes variations de la valeur moyenne. Nous avons tracé la droite de régression obtenue à partir des valeurs moyennes par taille d'instances. La valeur du coefficient de corrélation ($r_{xy} \approx -0.566$) confirme cette tendance, mais la valeur absolue de r_{xy} – bien inférieure à 1 – reflète les fortes variations détectées sur la courbe.

Par conséquent, en utilisant uniquement l'AIRLNDM(f_3), il est difficile d'exprimer des conclusions claires sur la difficulté des instances du *jobshop* simple même si des tendances se dégagent de cette étude.

Contrairement aux résultats de [MB96b] (section 7.1.3, page 86), la longueur de marche de l'AIRLNDM(f_3) ne semble pas constituer un bon indicateur de difficulté. Il suffit pour s'en convaincre d'observer les résultats obtenus par l'ensemble des instances de taille 10x10 dans le tableau A.3 (page 167) : sur 18 instances de taille 10x10, le nombre de pas moyen est de l'ordre de 25 avec un écart-type de l'ordre de 4 alors que les résultats de l'AIRLNDM(f_3) varient de 7.59% à 21.19% de l'optimum. De nouveau, même si une certaine tendance semble se dégager, la longueur de marche ne peut être utilisée comme indicateur de difficulté compte tenu de l'opérateur utilisé.

Les résultats obtenus à la fois pour les critères calculés sur les instances et pour l'AIRLNDM(f_3) ne permettent pas de définir un indicateur de difficulté pertinent. Cependant l'utilisation de plusieurs critères ou le recoupement des résultats obtenus par diverses méthodes de recherche permettent de distinguer certaines instances faciles à résoudre et d'autres instances difficiles à résoudre de manière quasi-indépendante de la méthode de recherche utilisée.

Ainsi par exemple, l'instance la29 considérée comme particulièrement difficile par différents auteurs, est effectivement classée en dernier pour les instances de taille 20x10 en ce qui concerne les performances de notre AIRLNDM(f_3).

De manière plus générale, les instances de Lawrence la21, la24, la25, la27, la29, la38 et la40 sont considérées comme des instances très difficiles. À cette liste viennent s'ajouter un ensemble de six instances réputées difficiles [CL95a] : ft10, la02, la19, la36, la37,la39.

Inst	JxM	$M\log(J!)$	rang	pos
la02	10x5	76	5/5	28/79
ft10	10x10	151	12/18	47/79
la19	10x10	151	7/18	32/79
la21	15x10	279	3/5	46/79
la24	15x10	279	4/5	50/79
la25	15x10	279	2/5	45/79
la36	15x15	418	1/15	42/79
la37	15x15	418	4/15	55/79
la38	15x15	418	15/15	79/79
la39	15x15	418	7/15	61/79
la40	15x15	418	10/15	66/79
la27	20x10	423	4/5	49/79
la29	20x10	423	5/5	72/79

Tableau 7.4: Classement des instances difficiles par l'AIRLNDM(f_3). Le rang indique la position de l'instance courante parmi les instances même taille. La position (notée pos) indique la place occupée par l'instance courante parmi les 79 premières instances classées dans le tableau A.3 (page 172).

Le tableau 7.4 tente d'établir une relation entre le classement des instances par rapport aux performances de l'AIRLNDM(f_3) et la difficulté des instances. Le classement par position permet de positionner 11 des 13 instances dans la seconde moitié du classement (les instances la02 et la19 figurent dans la première moitié du classement). Hormis pour la02, le tableau A.3 (page 172) indique que l'AIRLNDM(f_3) obtient un makespan à plus de 10% de l'optimum.

Ces résultats montrent que connaissant *a priori* un sous-ensemble d'instances difficiles, ces dernières peuvent être détectées dans une certaine mesure grâce aux performances relatives de l'AIRLNDM(f_3). Mais il s'agit davantage de tendance que d'indication fiable quant-à la difficulté, d'autant qu'il faut considérer un certain biais dû à la baisse de performance de l'AIRLNDM(f_3) en fonction de l'augmentation de la taille des instances. Ce biais traduit néanmoins une certaine corrélation entre la taille des instances et la difficulté. Ainsi, certains auteurs utilisent la taille de l'instance comme critère de difficulté. Par exemple, [JM99] considère qu'une instance telle que $J \times M \geq 200;\ J \geq 15;\ M \geq 10;\ J < 2.5 \times M$ est une instance difficile.

7.2.3 Discussion

Dans cette section, nous avons défini un certain nombre de critères en vue d'obtenir un indicateur de difficulté d'une instance conjointement à d'autres indicateurs « classiques » tels que le ratio $\frac{J}{M}$.

Nous avons introduit différents critères plus ou moins inspirés du flot de dominance et montré

expérimentalement que ces critères ne donnent pas d'indication pertinente sur la difficulté des instances. De même, de simples critères tels que la durée maximale u des opérations, ou le ratio $\frac{J}{M}$ donnent des tendances mais sont insuffisants pour mettre en évidence des familles d'instances pour lesquelles il sera facile d'obtenir une solution quasi-optimale. Par construction, le critère Δ_r (voir section 7.2.1, page 87) fournit une indication supplémentaire à ce propos : plus Δ_r est faible et plus il est facile d'obtenir une solution quasi-optimale. Là encore des tendances peuvent être dégagées pour un groupe d'instances, mais il est extrêmement hasardeux de prédire la difficulté à partir d'un seul de ces critères.

En fait, il est nécessaire pour aller plus loin d'étudier des associations de critères. Ceci complique l'étude à cause de l'interdépendance de certains critères (relation de dominances, combinaisons plus ou moins linéraires de critères, ...). Pour illustrer ces dépendances, nous pouvons utiliser une image qui consiste à restreindre le *jobshop* à la résolution simultanée de M problèmes de type sac-à-dos comportant chacun J objets : plus il y a d'objets (indiqué par le ratio $\frac{J}{M}$), plus les objets sont petits (indiqué par le critère u), plus les objets sont de tailles similaires (indiqué par les critères Δ_r et $\sigma_r\%$) plus il sera facile de résoudre en parallèle les sac-à-dos avec un taux de remplissage quasi-identique (ce qui correspond à un bon équilibrage de charge entre les machines, une fois ramené au problème de *jobshop*).

Pour obtenir l'AIRL-difficulté, notre calcul est basé sur le nombre de permutations effectuées par cet algorithme et non sur le nombre *minimum* d de permutations nécessaires pour transformer une solution de Λ en solution optimale. Nous avons également étudié d en terme de « distance-opérateur » (voir définition 6, page 16).

Le calcul de d n'est pas évident à effectuer : l'espace de recherche associé au *jobshop* comporte un grand nombre de dimensions et les distances « classiques » (distance de Manhattan $||\ ||_1$, distance Euclidienne $||\ ||_2$ ...) ne peuvent pas être utilisées. Pour des opérateurs simples, c'est-à-dire se limitant à effectuer des permutations d'opérations, il est possible de calculer une distance entre permutations, ou de définir des encadrements pour le calcul de distance entre des codages plus élaborés tels que les tableaux de marqueurs.

Les opérateurs effectivement utilisés pour résoudre le jobshop intègrent des mécanismes de tassement par décalage-gauche, ou détectent les opérations critiques, si bien qu'il devient impossible de calculer d pour ces opérateurs. Au cours de nos recherches, nous avons étudié une distance basée sur une représentation des ordonnancements sous forme de graphes disjonctifs. Cette distance est obtenue en définissant un ordre sur les paires d'arcs disjonctifs, auxquelles une suite binaire est associée. À chaque paire d'arcs disjonctifs est associé un bit dont la valeur est définie par l'arbitrage de chaque paire de disjonction : un 1 représentant l'arc dans un sens, un 0 représentant le sens inverse. Pour calculer la distance entre deux ordonnancements P_1 et P_2, il suffit d'associer à chaque ordonnancement une chaîne de $MJ\frac{J-1}{2}$ bits, obtenue selon la méthode exposée ci-dessus, et de calculer une distance de Hamming entre les chaînes de bits.

Nous avons finalement renoncé à utiliser cette distance : d'une part, elle ne permet pas distinguer les opérations critiques des opérations non critiques, ce qui pose problème lorsque l'opérateur agit exclusivement sur les opérations critiques. D'autre part, dans le cadre de nos travaux l'utilisation essentielle d'une telle distance se limite aux mesures de corrélations coût-distance. Cette mesure fait intervenir la notion de distance à une solution optimale, or pour le JSP il y a en général plusieurs optima[16] et rien ne garantit que nous nous soyons dirigés vers l'optimum le plus proche lors de la

[16]Il suffit de permuter des opérations non critiques si l'opérateur le permet. Dans le cas contraire il peut exister plusieurs optima si des opérations critiques possèdent la même durée, auquel cas leur permutation peut ne pas modifier la valeur du makespan. Il peut également exister plusieurs solutions optimales s'il existe plusieurs chemins critiques, ou de manière plus générale lorsque la somme des durées de certaines opérations correspond à la somme des durées

résolution de l'instance considérée. L'existence de plusieurs optima complique le calcul de la distance (définie en terme de nombre minimum d'applications de l'opérateur pour transformer une solution en une solution optimale) et nous a poussé à définir une notion plus simple : l'AIRL-difficulté.

Les bornes inférieures et supérieures, ainsi que les algorithmes approchés nous donnent une indication sur la difficulté de l'instance. Les calculs de corrélation entre les différents critères mettent en évidence des instances atypiques que nous écarterons de nos tests (hormis les problèmes de Fisher et Thompson qui font bien souvent office de référence lors de comparaisons avec différents auteurs).

Ces informations mettent en évidence les instances difficiles sur lesquelles nous devons concentrer nos efforts pour obtenir des résultats représentatifs en considérant un minimum d'instances. Cependant, compte tenu de l'antériorité de certains de nos travaux par rapport à la définition de la difficulté, les instances choisies pour effectuer les mesures dans la suite du mémoire ne sont pas nécessairement les plus *faciles* ou les plus *difficiles*. Elles ont été choisies de manière à être représentatives en terme de fréquence d'utilisation par d'autres auteurs (ft10, ft20) ou en terme de taille d'instance (ta01, abz7, ta31) après avoir écarté un certain nombre d'instances jugées « atypiques » lors de tests préliminaires. À présent, la définition de la difficulté des instances nous permet de prédire – dans une certaine mesure – cette notion d'instance « atypique », voire de l'expliquer dans certains cas extrêmes (présence d'une machine goulot, $\frac{J}{M} \gg 5$, durée des opérations quasi-identiques, ...).

7.3 Fonctions coût multicritères

Les algorithmes itératifs de recherche locale (AIRL) fondent principalement leurs décisions sur le coût du point courant et des points voisins. Le choix du voisin sélectionné pour l'itération suivante n'est déjà pas chose facile, il s'avère encore plus difficile lorsque les voisins ont la même valeur de coût. C'est-à-dire lorsque l'AIRL se trouve sur un plateau.

La résolution de problèmes d'optimisation combinatoire consiste typiquement à optimiser une quantité que nous appelons le critère principal (longueur du tour pour le problème du voyageur de commerce, durée globale de production dans les problèmes d'ordonnancement par exemple). La plupart du temps, lorsque ces problèmes sont résolus par des AIRL (soit de simples AIRL, soit des AIRL plus sophistiqués tels que la recherche tabou ou les algorithmes évolutifs), la fonction coût utilisée pour choisir le prochain point à visiter est uniquement basée sur le critère principal. Pour des problèmes classiques, tels que le voyageur de commerce (TSP) ou le problème du *jobshop* (JSP), nous pouvons mettre en évidence un certain nombre de « bonnes propriétés » – mesurées par des critères secondaires – qui caractérisent une solution de bonne qualité. Apparemment redondants en regard du critère principal, ces critères secondaires sont utiles lorsque le critère principal ne permet pas de distinguer des solutions au voisinage du point courant. Ainsi, lorsque l'AIRL est sur un plateau (aucun voisin n'est meilleur que le point courant du point de vue de la valeur de critère principal), ces critères secondaires peuvent aider à choisir le prochain point à visiter. Cette technique peut également être utilisée pour s'échapper d'un optimum local : lorsque le critère principal ne peut plus être amélioré, les critères secondaires peuvent alors être utilisés pour guider temporairement l'AIRL. Bien évidemment, les critères secondaires doivent être pertinents et apporter des précisions quant-à la valeur du critère principal. D'une certaine manière, ils doivent être plus précis que le critère principal afin de distinguer des solutions qui possèdent la même valeur pour le critère principal.

Nous appliquons cette technique au *jobshop* simple car le makespan n'est pas un critère suffisant pour comparer des ordonnancements entre eux : en effet, un grand nombre d'ordonnancements différents peuvent correspondre à une même valeur de makespan. Une fonction uniquement basée sur

d'autres opérations.

la valeur du makespan est incapable de « guider » efficacement la méthode utilisée dans sa recherche de l'optimum. C'est pourquoi, il est nécessaire de définir plusieurs critères complémentaires afin de comparer des ordonnancements entre eux.

Différentes techniques peuvent être utilisées pour prendre en compte ces critères, comme par exemple un mécanisme basé sur la méthode de Pareto [HN93, FF93, VFM95, SF96, Lan95, Kur90, TMK97, VRUC97, LR97] ou la coévolution de plusieurs espèces dans un algorithme évolutif [Hus94]. La technique d'intégration de critères supplémentaires dans la fonction coût se rencontre souvent dans les problèmes où existe un grand nombre de contraintes [RPLH89, SS89, EvdH97, CDM90, BEW95, SM95, PS93, IKB+97], surtout lorsque ces contraintes peuvent être réparties selon plusieurs niveaux de priorité. Dans ce cas, une partie (ou la totalité) des contraintes est reportée dans la fonction coût sous forme d'un mécanisme de pénalités statiques ou dynamiques [RPLH89, DG94, DBB94, ERR94, LS94, Jul93, ST93, vKHHK95, BE95, HK94, RCF94a, JM91b, Mic96, JH, MDRS96, MT96, Maq96, FGL96, Mic95, MA94, Gil97, KP98]. Cette méthode permet de simplifier le codage et les opérateurs utilisés. Le mécanisme de pénalités peut également être utilisé pour moduler l'importance relative de plusieurs critères à optimiser simultanément [HH97, Mic96].

Bien que basées sur un principe similaire, nous définissons des fonctions multicritères pour un tout autre objectif : il s'agit de guider la méthode de recherche durant l'optimisation d'un critère principal en ajoutant des critères secondaires et non de trouver le « meilleur compromis possible » entre les différents critères à optimiser. Dans les fonctions utilisées, l'objectif demeure l'optimisation du critère principal C, et c'est uniquement entre des solutions de même valeur de C que les autres critères interviennent. Aussi, nous avons proposé une technique permettant de guider la méthode de recherche sur ces plateaux lors de la résolution de problèmes $\mathcal{NP}$-durs [DPF+97, DPF+98, DPT+98]. Il faut alors faire la distinction entre la fonction objectif (utilisée pour guider la méthode de recherche) et fonction coût (coût effectif de la solution utilisé pour évaluer sa qualité en tant que solution au problème posé).

Nous définissons un *plateau* de la manière suivante :

Définition 60 (plateau)
Soient $s \in \mathcal{E}$ un point de l'espace de recherche, et v une valeur prise dans l'intervalle des valeurs du critère C. Considérons $X \subset V_{\mathcal{O}}(s)$ l'ensemble des points s' (voisins de s) tels que $C(s') = v$. X est un plateau s'il comporte au moins 2 éléments.

Par extension, le coût d'un plateau est donné par le coût de l'un des points qui le constituent : $C(X) = C(s') = v$.

L'idée consistant à utiliser une fonction auxiliaire pour guider la méthode de recherche est évoquée dans [Glo95]. L'utilisation de mesures heuristiques dans une fonction coût afin d'augmenter la précision lors de la comparaison de chemins pour un problème de robotique est évoqué dans [Mic96]. Un principe similaire est utilisé pour résoudre le *jobshop* dans [HW95], mais c'est la première fois, à notre connaissance, que de telles fonctions coût sont utilisées pour résoudre le *jobshop* simple.

L'ajout de critères secondaires dans la fonction coût permet de mieux guider la méthode dans l'espace de recherche. Aussi nous comparons plusieurs possibilités pour voir si cette amélioration de la fonction coût permet d'utiliser des opérateurs simples :

- opérateur simple non dédié au problème traité et fonction coût multicritère ;
- opérateur dédié au problème traité et fonction coût simple ;
- opérateur dédié au problème traité et fonction coût multicritère.

Dans cette section, nous étudions divers critères secondaires susceptibles d'être utilisés pour améliorer le choix du voisin à visiter. Ainsi que nous le suggérons dans [DPF+98, DPT+98], nous montrons que la prise en compte de ces informations améliore les résultats de manière significative. Afin de mieux comprendre le fonctionnement des méthodes de recherches dans le paysage associé au *jobshop*, nous développons notre idée en utilisant un simple AIRL déterministe (AIRLD) basé sur un opérateur de déplacement similaire à l'opérateur représenté par la figure 7.2 (page 82).

7.3.1 Évolution sur les plateaux

Les plateaux peuvent avoir plusieurs origines :

- redondances du codage ;
- opérateurs susceptibles de se ramener à l'identité sous certaines conditions ;
- passage du codage indirect à l'individu (génotype $\longmapsto$ phénotype) ;
- manipulations d'informations non pertinentes (dont l'influence n'est pas immédiatement perceptible dans la fonction coût, pour le JSP c'est le cas des opérations non critiques ou lorsqu'il existe plusieurs chemins critiques en parallèle) ;
- fonction coût imprécise ;
- mécanisme de réparation (cas des codages autorisant des solutions invalides).

Le critère $\varphi_\sharp$, défini lors de l'étude des fonctions coût multicritères, nous fournit des informations sur ces plateaux sans toutefois déceler leur origine.

Pour éliminer les cycles courts (sur les plateaux), nous utilisons une mémoire de mouvements. L'idée de doter une méthode de recherche amélioratrice de mémoire, afin de la guider dans l'espace de recherche, n'est pas nouvelle : le succès de la recherche tabou repose largement sur ce principe.

Dans le cadre de notre étude de méthodes de recherche simples, nous comparons entre eux différents AIRL, plus ou moins dotés de mémoire.

7.3.1.1 Influence des plateaux

Il est à noter que la présence de plateaux dans l'espace de recherche n'est pas nécessairement synonyme de détérioration de performances. Tout dépend en fait du type de méthode de recherche utilisée. Ainsi une méthode de recherche qui choisit un voisin aléatoirement dans le voisinage a plus de chance de trouver un chemin vers l'optimum si l'espace de recherche comporte des plateaux.

Il n'est cependant pas évident de quantifier l'impact de la présence de plateaux sur les performances des méthodes à utiliser pour résoudre tel ou tel problème : le calcul du nombre et de la taille des plateaux n'est déjà pas chose facile, mais il faut également tenir compte de ce que l'on pourrait qualifier de « degré d'imbrication des plateaux ». Pour illustrer cette notion, nous donnons simplement un moyen d'estimer cette notion : pour chaque plateau, supposons que nous puissions calculer le nombre de plateaux p qui lui sont connexes ainsi que leurs altitudes respectives $A(i)$. En ce cas, plus p est grand, plus le nombre d'alternatives (passage d'un plateau à l'autre) est grand pour une méthode de recherche donnée. Plus il y a de valeurs différentes pour les $A(i)$[17] plus le paysage est « rugueux » et plus il y a de variations d'altitude possibles lors du passage d'un plateau à un autre.

Ces notions de rugosité ou de degré d'imbrication ne sont pas encore suffisantes pour déterminer quelle méthode utiliser en présence de plateaux. Elles permettent juste d'affirmer qu'il ne faut pas

[17] $A(i)$ est l'altitude du plateau i, voisin du plateau considéré à un instant donné

conclure que la présence de plateaux est synonyme de mauvaises performances. Pour argumenter en ce sens, nous pouvons concevoir deux espaces de recherche : E_1 dans lequel p et le nombre de valeurs $A(i)$ sont faibles, et E_2 dans lequel p et le nombre de valeurs $A(i)$ sont élevés. Ce qui va influencer le comportement des méthodes de recherche dans E_1 ou E_2 sera essentiellement la présence d'un ou de plusieurs optima, le nombre et la répartition des optima locaux dans l'espace, la taille des bassins d'attraction, la présence d'attracteurs vers des optima locaux et non les valeurs de p et $A(i)$ ni même la présence de plateaux.

7.3.1.2 Principe

Dans cette section, nous considérons les problèmes de minimisation[18] définis par :

Soient $\mathcal{E}$ un espace de recherche discret et un critère C défini dans $\mathcal{E}$, trouver une solution $x^* \in \mathcal{E}$ telle que $x^* = \min_{x \in \mathcal{E}}\{C(x)\}$.

Lors de l'utilisation d'un AIRL pour résoudre un tel problème, l'idée usuelle est de concevoir la fonction objective f de manière à renvoyer la valeur de la quantité à optimiser pour le point considéré x, c'est-à-dire $f(x) = C(x)$.

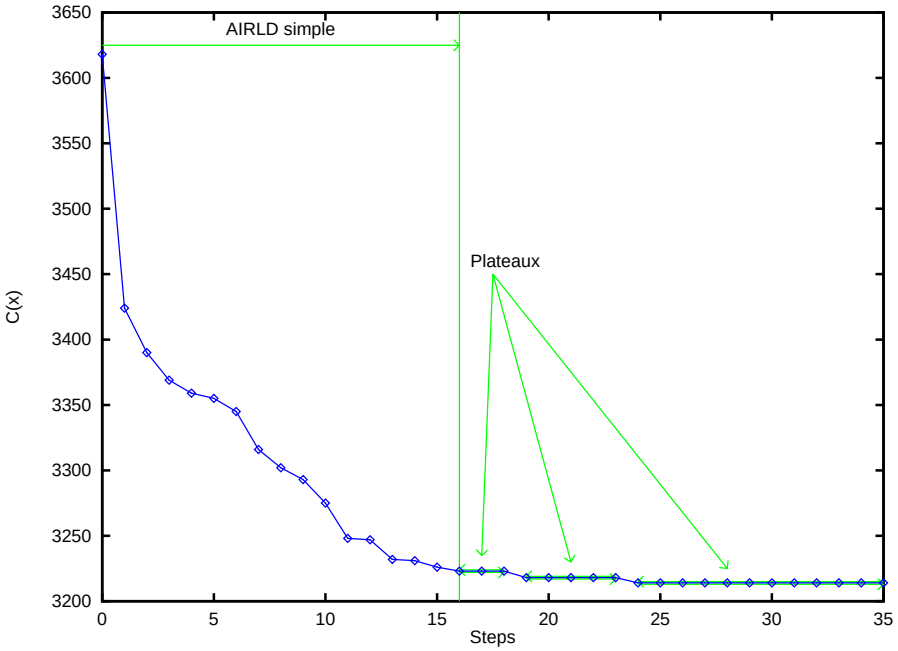


Figure 7.7: Évolution typique du critère à optimiser $C(x)$ au fil des itérations d'un AIRLDPM. L'AIRLD utilisé est modifié de manière à mettre en évidence les plateaux : la solution voisine retenue est x' tq $C(x') = \min_{s' \in V_{\mathcal{O}}(x)} C(s')$, le déplacement de x à x' est accepté même si $C(x') = C(x)$ et l'AIRL est doté d'une mémoire similaire à une liste Tabou (de taille ≥ 12 pour la courbe présentée) pour éliminer les cycles courts.

Nous avons tracé l'évolution typique de $C(x)$ au cours d'une exécution d'un AIRLDPM dont l'objectif est la minimisation de $C(x)$ (*cf* figure 7.7) : x peut être vu comme le point courant durant l'exécution d'un AIRLDPM ou d'une recherche tabou, ou encore comme le meilleur point de la population d'un AE. La courbe présente plusieurs plateaux. Sur chaque plateau, l'algorithme a des difficultés pour se guider dans le voisinage de la solution courante (voir figure 7.8). Dans cette situation, l'utilisation d'informations complémentaires peut s'avérer utile pour guider la recherche.

[18] Bien entendu, les résultats présentés sont transposables aux problèmes de maximisation.

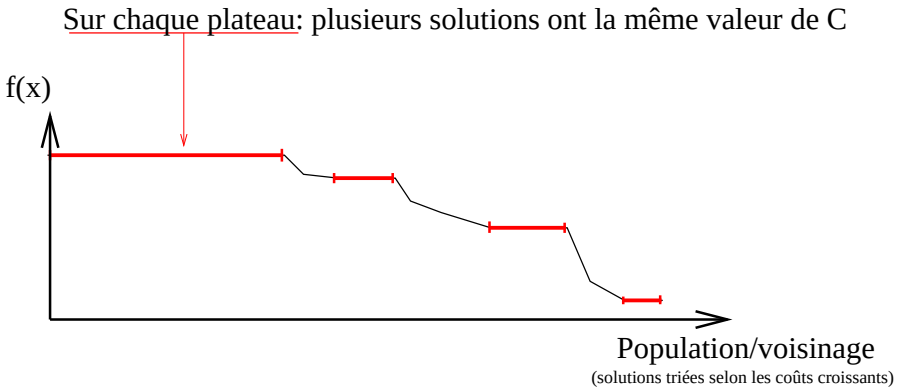


Figure 7.8: Mise en évidence des plateaux : sur chaque plateau l'AIRL n'est plus guidé dans le voisinage de la solution courante x, les voisins x' de coût minimum sont tels que $f(x') = f(x)$. f ne permet pas de distinguer une solution parmi les meilleures solutions voisines. Le critère principal C, utilisé seul dans la fonction coût ($f(x) = C(x)$) ne permet pas de discerner une solution parmi les meilleures solutions voisines.

Figure 7.9: Utilisation d'une fonction multicritère : des informations complémentaires sont ajoutées dans la fonction coût f' en plus du critère principal C (*i.e.* $f'(x) \neq C(x)$). Le principal objectif de l'intégration d'informations complémentaires est d'améliorer la fonction coût afin de mieux discerner des points composant le voisinage de la solution courante. Autrement dit, il s'agit de réduire artificiellement la taille des plateaux vus par l'AIRL au travers de f' en choisissant les informations complémentaires de manière à aboutir le plus fréquemment possible à la situation « idéale » suivante sur les plateaux de f :
Soient x', x'' tq $x' \neq x''$, si $f(x') = f(x'') = \min_{s' \in V_{\mathcal{O}}(x)} f(s')$ alors $f'(x') \neq f'(x'')$.
À défaut de pouvoir définir de telles informations complémentaires, la situation présentée ci-dessus peut être généralisée de manière à retenir les critères vérifiant la propriété suivante :
Soient x', x'' tq $x' \neq x''$, si $f(x') = f(x'')$ alors $f'(x') \neq f'(x'')$ avec une *forte probabilité*.
x est une solution courante voisine du plateau contenant x' et x''.

Ces informations complémentaires – ou critères secondaires – servent à discerner des points qui sont, en apparence, identiques (identiques du point de vue du critère principal $C(x)$) alors qu'ils sont réellement différents (voir figure 7.9).

De manière à simplifier l'étude de cette technique et à évaluer ses effets sans multiplier les paramètres, nous avons choisi un algorithme très simple : un algorithme itératif de recherche locale déterministe (sans mécanisme de déplacement sur les plateaux). Ce dernier comporte deux paramètres principaux : l'opérateur de déplacement et la fonction de sélection. L'opérateur retenu permute deux opérations et ré-ordonnance les opérations lorsque cela s'avère nécessaire (voir figure 7.2, page 82). L'utilisation de cet opérateur « simple » nous permet d'étudier la fonction de sélection par elle-même. La fonction coût est alors la seule information capable de guider la méthode dans l'espace de recherche.

Une fois la technique validée sur l'AIRLD, nous montrons qu'elle est applicable aux algorithmes itératifs de recherche locale non déterministes (AIRLND) et qu'elle améliore la qualité des solutions finales.

7.3.1.3 Pouvoir discriminant d'un critère

Un critère (secondaire) C' est utile s'il permet de discerner des points qui ont une même valeur pour le critère principal C. Ce « pouvoir discriminant » est une caractéristique très importante des critères secondaires, et il s'avère nécessaire d'en donner une définition appropriée.

Nous introduisons pour cela une valeur notée $\varphi_\sharp(C')$ qui caractérise le « pouvoir discriminant » d'un critère donné C' :

Définition 61 (pouvoir discriminant)
Le pouvoir discriminant du critère secondaire C', noté $\phi_\sharp(C')$, est le nombre de valeurs différentes prises par C' sur un plateau de C.

Le pouvoir discriminant moyen, noté $\varphi_\sharp(C')$, est obtenu en calculant la moyenne des $\phi_\sharp(C')$ sur un ensemble de solutions.

Un critère C' tel que $\varphi_\sharp(C') \approx 1$ n'est pas un critère intéressant puisque – en moyenne – il y a une seule valeur de C' sur les plateaux de C : cela signifie que C' ne peut pas être utilisé pour guider la méthode de recherche sur les plateaux de C.

7.3.2 Application au problème du Jobshop

Tournons nous maintenant vers l'application de cette technique au *jobshop* simple. Considérons une instance $J \times M$ du *jobshop* simple, et un ordonnancement x relatif à cette instance.

Il existe de très nombreux critères secondaires pour le *jobshop*, aussi avons nous choisi trois critères parmi ces derniers. Ces trois critères ne sont pas nécessairement les plus performants, ils permettent simplement de montrer sur un exemple concret que le principe de fonction multicritère est applicable au *jobshop*.

Dans une précédente section, nous avons montré l'importance des opérations critiques lors de la résolution du *jobshop*. Par conséquent, notre premier critère secondaire est le nombre d'opérations critiques. Il est noté $C_{\text{op}}(x)$. D'ailleurs, d'un point de vue pratique, il est préférable de minimiser le nombre d'opérations critiques car moins il y a d'opérations critiques, plus le nombre d'opérations qu'il est possible de retarder sans affecter le makespan est important. C'est pourquoi nous considérons que pour un ensemble d'ordonnancements possédant la même valeur de makespan, la qualité de l'ordonnancement est inversement proportionnelle au nombre d'opérations critiques.

Notre second critère, inspiré de [HW95, HW96], est noté $H_2(x)$: $H_2(x) = \sum_{m=1}^{M} (E^m(x))^2$

$E^m(x)$: date de fin de la dernière opération réalisée sur la machine M_m selon l'ordonnancement x.

Remarque : Le critère $H_2(x)$ permet de pénaliser les ordonnancements pour lesquels plusieurs machines sont utilisées jusqu'à une date proche du makespan.

Enfin notre dernier critère, noté $C_2(x)$ est défini par : $C_2(x) = \sum_{j=1}^{J} (E_j(x))^2$

$E_j(x)$: date de fin de la dernière opération du produit J_j réalisée selon l'ordonnancement x.

Remarque : Le critère $C_2(x)$ permet de pénaliser les ordonnancements pour lesquels la date de fin de la dernière opération de plusieurs produits est proche du makespan.

7.3.2.1 Fonction coût de base

Pour le *jobshop* simple, où l'objectif est la minimisation du makespan, l'idée usuelle consiste à utiliser le makespan comme critère principal (*i.e.* $C(x) = C_{\max}(x)$) pour définir la fonction coût par :

$$f_1(x) = C_{\max}(x)$$

Or, nous avons remarqué qu'une fonction coût uniquement basée sur le makespan est insuffisante pour guider efficacement l'algorithme dans l'espace de recherche, surtout lorsque les durées des opérations de l'instance considérée sont proches. Ceci est dû au fait qu'un grand nombre d'ordonnancements distincts ont une même valeur de makespan. Aussi avons nous défini une fonction coût, basée principalement sur le makespan, utilisant des critères secondaires chargés de discriminer ces ordonnancements [DPF+97, DPF+98, DPT+98].

7.3.2.2 Pouvoir discriminant de différents critères pour le *jobshop*

Bien que de nombreux critères puissent être utilisés comme indicateurs de la qualité d'une solution[19], nous exposons uniquement les résultats des critères H_2 et C_{op} dans le but de montrer la validité de notre approche (sans chercher les meilleurs critères utilisables).

Nous calculons la valeur de $\varphi_\sharp$ pour les deux critères étudiés le long des marches de l'AIRLD : le tableau 7.5 donne la valeur moyenne et l'écart-type de $\phi_\sharp$ pour les deux critères le long des marches de l'AIRLD.

La valeur de $\varphi_\sharp$ est supérieure à un pour les deux critères étudiés. Cela signifie que ces critères sont en mesure de discerner des points sur un plateau du makespan. Ils sont donc susceptibles d'être utiles dans la fonction coût. De plus, $\varphi_\sharp(H_2)$ est toujours supérieur à $\varphi_\sharp(C_{\text{op}})$, ce qui signifie que le pouvoir discriminant de H_2 est supérieur à celui de C_{op}.

7.3.2.3 Corrélation entre critères

La condition $\varphi_\sharp > 1$ ne suffit pas pour déterminer si un critère secondaire permet de guider efficacement la méthode de recherche : il faut vérifier si la variation de ce critère est corrélée à celle du makespan.

[19] Comme par exemple le critère C_2 (voir section 7.3.2 page 104)

Instance		$\phi_\sharp(H_2)$		$\phi_\sharp(C_{\text{op}})$	
Nom	$J \times M$	Moyenne ($\varphi_\sharp$)	Écart-type	Moyenne ($\varphi_\sharp$)	Écart-type
ft10	10×10	3.92	0.56	2.60	0.34
ft20	20×5	6.39	1.16	4.69	0.88
ta01	15×15	6.85	1.31	3.89	0.75
abz7	20×15	27.69	5.93	20.53	4.98
ta31	30×15	33.19	6.67	23.03	5.29

Tableau 7.5: Valeur moyenne et écart-type de $\phi_\sharp$ pour différents critères le long des marches de l'AIRLD. Pour chaque instance, 500 solutions initiales sont générées aléatoirement. L'AIRLD est ensuite exécuté à partir de ces solutions. La fonction coût utilisée par l'AIRLD est f_1.

	Instance		Cmax/H2	Cmax/C2	Cmax/Cop	H2/Cop	C2/Cop
Solutions initiales	ft10	10×10	0.915	0.859	0.591	0.565	0.540
	ft20	20×5	0.790	0.847	0.408	0.313	0.425
	ta01	15×15	0.835	0.656	0.383	0.365	0.252
	abz7	20×15	0.831	0.701	0.249	0.191	0.165
	ta31	30×15	0.870	0.673	0.290	0.251	0.230
	Moyenne		0.848	0.747	0.384	0.337	0.322
Optima locaux	ft10	10×10	0.854	0.725	0.132	0.118	0.155
	ft20	20×5	0.872	0.839	0.144	0.127	0.158
	ta01	15×15	0.870	0.806	0.206	0.183	0.178
	abz7	20×15	0.847	0.781	0.060	0.030	0.026
	ta31	30×15	0.889	0.710	0.159	0.092	0.015
	Moyenne		0.866	0.772	0.140	0.110	0.106

Tableau 7.6: Valeurs moyennes des corrélations entre différents critères pour chaque instance étudiée. La corrélation est calculée sur un ensemble de 500 solutions générées aléatoirement et sur un ensemble d'optima locaux obtenu par l'AIRLD. L'AIRLD est lancé à partir de chaque solution aléatoire, il utilise la fonction coût (f_1) basée exclusivement sur le makespan. De cette façon, nous obtenons au plus 500 optima locaux.

De manière à déterminer quel critère est susceptible d'être utilisé dans la fonction coût, nous avons calculé la corrélation entre chaque critère et le makespan. Ainsi, nous savons quel critère est corrélé à la qualité d'un point (en terme de makespan).

Les résultats sont donnés dans le tableau 7.6 : les résultats montrent que la corrélation entre $C_{\max}$ et H_2 est très élevée, alors que la corrélation entre $C_{\max}$ et C_{op} est plutôt faible pour les solutions aléatoires (quasi-inexistante lorsque l'on considère les optima locaux). Donc, selon ces valeurs, H_2 semble être un critère intéressant à intégrer dans notre fonction coût. En revanche, C_{op} ne semble donner que très peu d'informations complémentaires lorsqu'il s'agit de guider la méthode de recherche.

Ces résultats montrent une bonne corrélation linéaire entre $C_{\max}$ et H_2, or le makespan évolue en fonction de E^m et H_2 en fonction de $(E^m)^2$. En conséquence, nous avons étudié la corrélation entre $C_{\max}$ et la $\sqrt{C_2}$ et $\sqrt{H_2}$. Les valeurs obtenues sont similaires aux résultats présentés dans le tableau 7.6.

L'étude des corrélations instance par instance met en évidence un comportement atypique pour les problèmes de Fisher et Thompson : les plus fortes corrélations entre les différents critères pour les solutions initiales sont associées à ft10. De même, les plus fortes corrélations (hormis ft10) Cmax/C2, Cmax/Cop et C2/Cop et la plus faible corrélation Cmax/H2 pour les solutions initiales sont associées à ft20. La plus forte corrélation Cmax/C2 pour les optima locaux est associée à ft20. Enfin, ft20 correspond à la seconde valeur la plus élevée en ce qui concerne les corrélations Cmax/H2, H2/Cop et C2/Cop pour les optima locaux. L'instance abz7 obtient quasi-systématiquement les plus faibles corrélations pour Cmax/Cop, H2/Cop et C2/Cop.

En ce qui concerne l'instance ft20, les résultats relatifs aux corrélations impliquant les critères C_2 et H_2 ne peuvent s'expliquer totalement par le nombre peu élevé de produits (J=5) et le ratio $\frac{J}{M}$ important. La suite de notre étude tend à confirmer l'atypie des instances ft10 et ft20.

7.3.2.4 Quelques fonctions coût

Dans la section précédente, nous avons introduit plusieurs critères qui semblent être de bons candidats à l'intégration dans la fonction coût. Nous définissons maintenant plusieurs fonctions coût à partir de ces critères afin d'évaluer leur utilité effective lors de la résolution du *jobshop*. Dans les fonctions suivantes, nous définissons k_i (où $1 \leq i \leq 3$) de manière à ce que le makespan reste le critère le plus important (critère principal) : une variation du makespan est toujours supérieure à la somme des variations des autres critères (critères secondaires) pour tous les ordonnancements sans-délai de l'instance de *jobshop* considérée.

- Tout d'abord, nous pouvons intégrer le critère H_2 dans la fonction coût. Nous appelons cette fonction $f_2(x)$:

$$f_2(x) = k_1 \times C_{\max}(x) + H_2(x)$$

k_1 est une borne supérieure pour $H_2(x)$ sur tous les ordonnancements sans-délai de l'instance de *jobshop* considérée.

- Nous avons également testé le critère C_2, similaire à H_2, dans la fonction $f_6(x)$:

$$f_6(x) = k_4 \times C_{\max}(x) + C_2(x)$$

k_4 est une borne supérieure pour $C_2(x)$ sur tous les ordonnancements sans-délai de l'instance de *jobshop* considérée.

• Ainsi que cela a été précédemment énoncé, le nombre d'opérations critiques peut être utilisé comme indicateur de la qualité d'un ordonnancement. Par conséquent, nous définissons une fonction coût basée sur ce critère. Nous appelons cette fonction $f_5(x)$:

$$f_5(x) = k_2 \times C_{\max}(x) + C_{\text{op}}(x)$$

k_2 est une borne supérieure pour $C_{\text{op}}(x)$ sur tous les ordonnancements sans-délai de l'instance considérée.

• Afin d'évaluer la pertinence des informations ajoutées, nous introduisons une fonction où le critère ajouté est remplacé par du bruit. Nous appelons cette fonction $f_4(x)$:

$$f_4(x) = k_3 \times C_{\max}(x) + \text{Random}(k_3)$$

Random(k_3) est un générateur de nombres aléatoires (distribution uniforme) qui retourne des nombres entiers compris entre 0 et $k_3 - 1$.

• Nous pouvons également combiner les deux critères étudiés au sein d'une même fonction. Ainsi, nous utilisons une fonction coût qui combine le makespan, le critère H_2 et le nombre d'opérations critiques. Nous appelons cette fonction $f_3(x)$:

$$f_3(x) = k_2 \times k_1 \times C_{\max}(x) + k_2 \times H_2(x) + C_{\text{op}}(x)$$

• Dans toutes les fonctions f_1 à f_5, le makespan est le critère principal. Finalement, nous définissons une fonction dans laquelle le critère principal est H_2. Nous appelons cette fonction $f_0(x)$:

$$f_0(x) = k_0 \times H_2(x) + C_{\max}(x)$$

k_0 est une borne supérieure pour $C_{\max}(x)$ sur tous les ordonnancements sans-délai de l'instance considérée.

Pour la suite de l'étude, nous avons fixé les coefficients k_i à la valeur maximale de chaque critère :

- $k_0 = D+1$ où $D = C^u_{\max}$ est la somme des durées de toutes les opérations de l'instance considérée : il est facile de montrer que le makespan des ordonnancements sans-délai est toujours inférieur à cette valeur.
- $k_1 = M \times D^2$ cette valeur est une conséquence immédiate de la valeur utilisée comme borne supérieure du makespan ci-dessus.
- $k_2 = J \times M$: il y a $J \times M$ opérations dans l'instance du *jobshop*, par conséquent il ne peut y avoir plus de $J \times M$ opérations critiques.
- $k_3 = k_0$: en utilisant cette valeur, les nombres calculés par Random(k_3) sont du même ordre de grandeur que le makespan.
- $k_4 = J \times D^2$ cette valeur est une conséquence immédiate de la valeur utilisée comme borne supérieure du makespan ci-dessus.

En fait, il est possible de définir ces valeurs plus précisément, mais cela est inutile pour notre utilisation : nous avons besoin de bornes supérieures, même imprécises, afin que le makespan demeure le critère principal.

7.3.2.5 Opérateurs et fonctions coût

Les études axées sur la fonction coût sont plutôt rares : en général, cette fonction est directement la valeur que l'on cherche à optimiser. Cependant, pour de nombreux problèmes, il existe des informations complémentaires capables de guider la méthode utilisée dans sa recherche d'une solution. Partant de cette constatation, nous étudions l'impact de l'ajout de critères secondaires dans la fonction coût. L'objectif consiste à modifier le paysage de manière bénéfique pour une méthode de recherche en utilisant des fonctions multicritères.

Les critères secondaires introduits dans nos différentes fonctions coût permettent de réduire le nombre de solutions voisines (obtenues par une application de l'opérateur à la solution courante) possédant un même coût. Cependant ces fonctions ne suppriment pas tous les plateaux[20] dans l'espace de recherche.

Nous cherchons plus particulièrement à déterminer si les opérateurs utilisés conjointement avec ces fonctions peuvent être simplifiés sans détériorer les performances des méthodes de recherche. Dans ce but, nous comparons deux familles d'algorithmes constituées à partir des composants suivant :

- opérateurs simples associés à des fonctions coût basées exclusivement sur le makespan ou multicritères ;
- opérateurs complexes associés à des fonctions coût basées exclusivement sur le makespan ou multicritères.

L'idée sous-jacente est la suivante : les performances obtenues par des opérateurs simples et une fonction multicritère sont-elles supérieures aux performances obtenues par des opérateurs complexes (éventuellement associés à un voisinage restreint) ?

Nous évaluerons expérimentalement cette idée dans la section dédiée aux résultats.

7.3.3 Résultats

Nous utilisons les instances de la bibliothèque *OR-Library* [Bea90]. Dans notre suite de tests, la taille des instances varie de 10×10 à 30×15.

7.3.3.1 AIRLD

L'AIRLD a été testé en utilisant les fonctions coût f_0 à f_6. Pour chaque instance, un ensemble de 500 solutions initiales est généré aléatoirement. Ensuite, les fonctions sont testées sur ce même ensemble. Les résultats obtenus sont présentés dans le tableau 7.7.

Pour chaque instance de notre suite de tests, f_3 obtient les meilleurs résultats en terme de qualité des optima locaux trouvés. Clairement, plus il y a d'informations ajoutées dans la fonction coût, meilleurs sont les résultats. Les plus mauvais résultats sont en général obtenus par la fonction initiale f_1 (où $C_{\max}$ est le seul critère).

La fonction f_4, qui intègre un générateur de bruit engendre de meilleurs résultats moyens que f_1.

La fonction f_0 où $C_{\max}$ n'est pas le critère principal obtient des résultats similaires à f_1. Les résultats obtenus par la fonction f_2 sont meilleurs que f_1 et f_0, ce qui tend à montrer que H_2 est utile en tant que critère secondaire mais pas en tant que critère principal.

[20] Ensemble de solutions voisines possédant un même coût (voir définition 60, page 99).

Instance	ft10	ft20	ta01	abz7	ta31
Taille	10x10	20x5	15x15	20x15	30x15
Optimum	930	1165	1231	656	1764
f_0 Moyenne	1073	1362	1465	779	2156
Écart-type	40	44	42	17	45
Minimum	958 (3.0)	1238 (6.3)	1371 (11.4)	735 (12.0)	1999 (13.3)
Maximum	1219	1515	1605	846	2305
f_1 Moyenne	1085	1351	1467	782	2150
Écart-type	39	46	46	18	43
Minimum	991 (6.6)	1228 (5.4)	1344 (9.2)	721 (9.9)	2036 (15.4)
Maximum	1214	1506	1626	853	2300
f_2 Moyenne	1069	1331	1440	770	2119
Écart-type	37	42	44	18	45
Minimum	974 (4.7)	1229 (5.5)	1338 (8.7)	725 (10.5)	1990 (12.7)
Maximum	1205	1452	1609	839	2278
f_3 Moyenne	1068	1328	1439	770	2118
Écart-type	37	43	45	19	44
Minimum	974 (4.7)	1229 (5.5)	1337 (8.6)	727 (10.8)	1990 (12.7)
Maximum	1205	1452	1609	839	2278
f_4 Moyenne	1083	1349	1459	782	2150
Écart-type	41	45	44	18	44
Minimum	971 (4.4)	1217 (4.5)	1352 (9.8)	729 (11.1)	2036 (15.4)
Maximum	1207	1488	1632	861	2300
f_5 Moyenne	1081	1347	1464	778	2144
Écart-type	39	46	46	19	43
Minimum	971 (4.4)	1222 (4.9)	1344 (9.2)	728 (11.0)	2021 (14.5)
Maximum	1214	1506	1626	853	2295
f_6 Moyenne	1070	1314	1443	770	2123
Écart-type	38	43	44	18	44
Minimum	974 (4.7)	1198 (2.8)	1333 (8.3)	724 (10.4)	1976 (12.0)
Maximum	1205	1440	1626	839	2280

Tableau 7.7: Résultats de l'AIRLD : les instances sont désignées par le nom défini dans la *OR-Library*. La deuxième ligne indique la taille de l'instance. La troisième ligne donne la valeur de l'optimum, ou un encadrement lorsqu'il est inconnu. Ensuite pour chaque fonction, le tableau contient la valeur moyenne, l'écart-type, la valeur minimale et la valeur maximale du makespan obtenus sur les 500 exécutions de l'AIRLD. Les valeurs entre parenthèses sont exprimées en pourcentage par rapport à la valeur optimale.

Nous pouvons établir un classement entre les différentes fonctions coût selon les performances moyennes :

$$f_3 > f_2 \geq f_6 > f_5 > f_4 \geq f_0 \geq f_1$$

où $f_3 > f_2$ signifie que f_3 donne de meilleurs résultats par rapport à f_2.

Si l'on considère les performances moyennes, le gain sur les cinq instances considérées est de l'ordre de 1.84% entre AIRLD(f_1) et AIRLD(f_2) et de l'ordre de 1.94% entre AIRLD(f_1) et AIRLD(f_3).

Un même classement pourrait être effectué pour les performances minimales, mais compte tenu du nombre d'expériences lancées face à la taille de l'espace de recherche, l'échantillonnage des solutions initiales influe sur la valeur du makespan de la meilleure solution obtenue : le fait que l'AIRL[21] trouve telle ou telle valeur de makespan minimum peut être un « coup de chance ». Par conséquent, nous comparons directement les résultats en terme de répartition des valeurs de makespan des solutions finales de l'AIRLD pour les fonctions étudiées (voir table 7.10). Nous établissons cette comparaison à partir des courbes, en considérant qu'une fonction donnant un plus grand pourcentage de solutions (ordonnée élevée) proche de l'optimum (abscisse faible) est supérieure (en terme de performance) à une autre fonction offrant un pourcentage plus faible de solutions ou une valeur plus éloignée de l'optimum. Par exemple, sur le schéma 7.10(d), la courbe (C_1) est relative à une fonction jugée plus performante que (C_2) : bien que fournissant pratiquement le même pourcentage de solutions proches de l'optimum, à qualité de solution égale (*i.e.* même abscisse) (C_2) donne très rapidement des pourcentages de solutions plus élevés que (C_1). Le classement pour ces trois courbes serait : $(C_1) > (C_2) > (C_3)$.

Nous avons placé sous les courbes $(C_1), (C_2), (C_3)$ les courbes correspondant aux pourcentages de solutions (non cumulés). Ces courbes montrent que (C_1) correspond à la répartition possédant à la fois la plus faible moyenne et le plus faible écart-type (gage de robustesse et de stabilité de la fonction utilisée dans l'AIRL).

Nous obtenons le classement des fonctions utilisées dans l'AIRLD selon la répartition des valeurs de makespan des solutions finales :

$$f_3 > f_2 \geq f_6 > f_5 > f_4 \geq f_1 > f_0$$

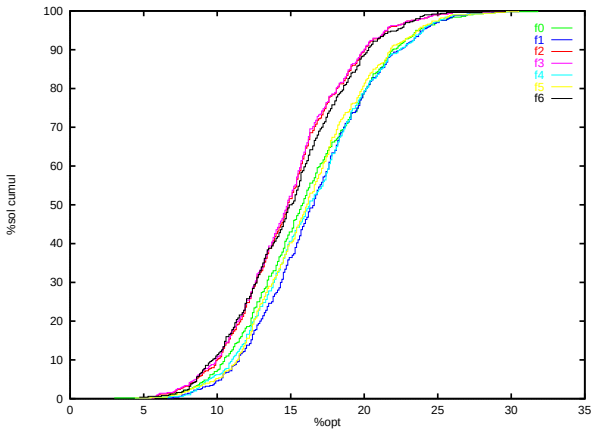
où $f_3 > f_2$ signifie que f_3 donne de meilleurs résultats par rapport à f_2.

Pour toutes les instances, la longueur de marche varie de manière significative d'une fonction coût à l'autre (*cf.* table 7.8).

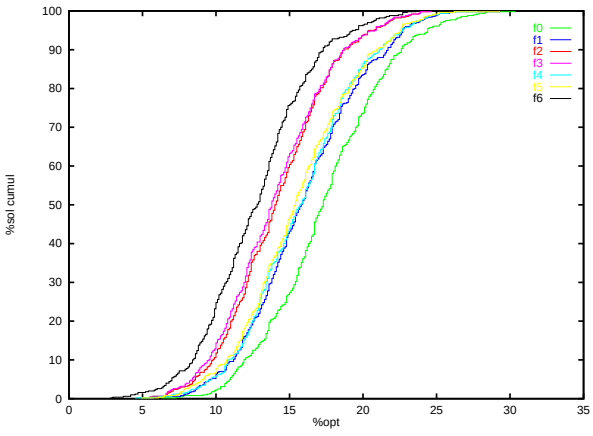
Instance	ft10	ft20	ta01	abz7	ta31
Taille	10x10	20x5	15x15	20x15	30x15
f_0	9 (3)	11 (4)	10 (4)	11 (4)	17 (6)
f_1	5 (2)	7 (3)	5 (2)	6 (3)	7 (3)
f_2	9 (4)	14 (6)	13 (6)	14 (6)	25 (10)
f_3	9 (4)	15 (6)	13 (6)	14 (6)	26 (11)
f_4	7 (3)	10 (4)	7 (3)	7 (3)	9 (3)
f_5	6 (3)	9 (4)	6 (3)	7 (3)	9 (4)
f_6	10 (4)	28 (10)	13 (6)	17 (7)	33 (12)

Tableau 7.8: Résultats de l'AIRLD : nombre moyen de pas nécessaires à la découverte d'un optimum local. Pour chaque instance, la moyenne est calculée sur 500 expériences, l'écart-type correspondant est indiqué entre parenthèses.

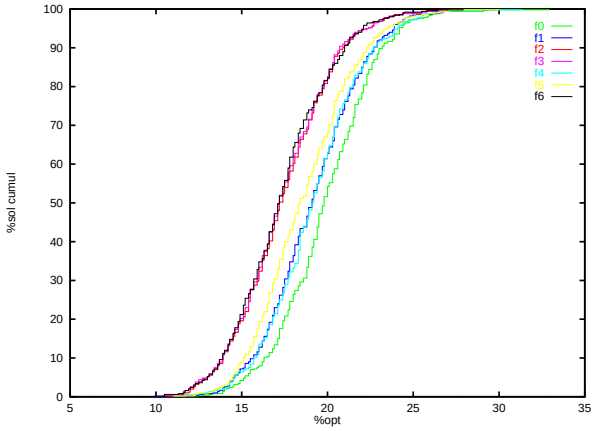
[21] Cette remarque est valable pour les AIRLD comme pour les AIRLND.



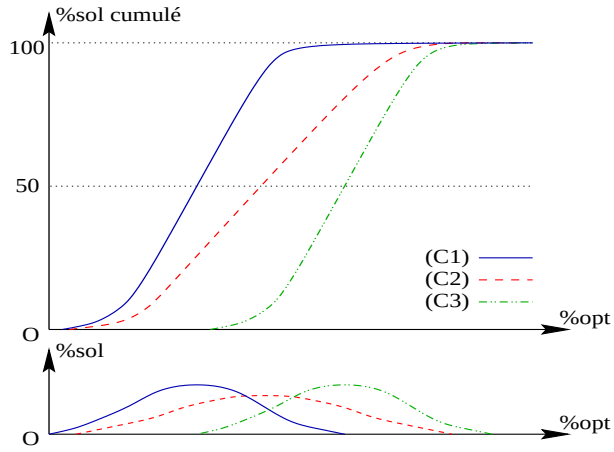
(a) ft10



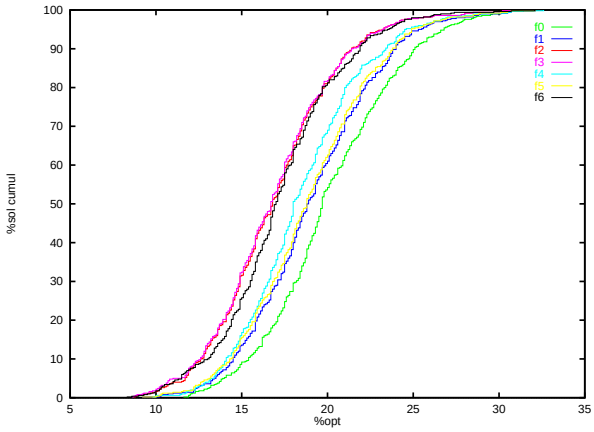
(b) ft20



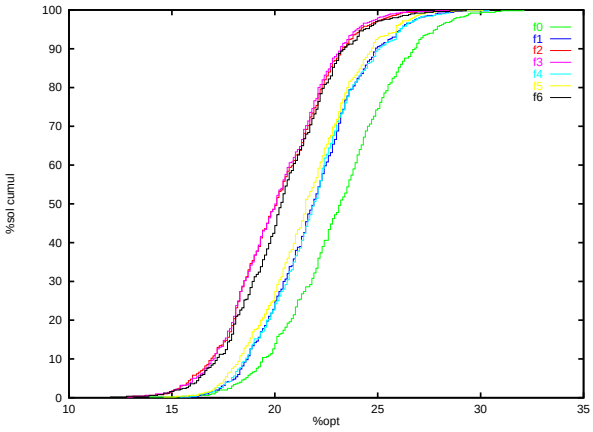
(c) abz7



(d) Relation entre % et % cumulé

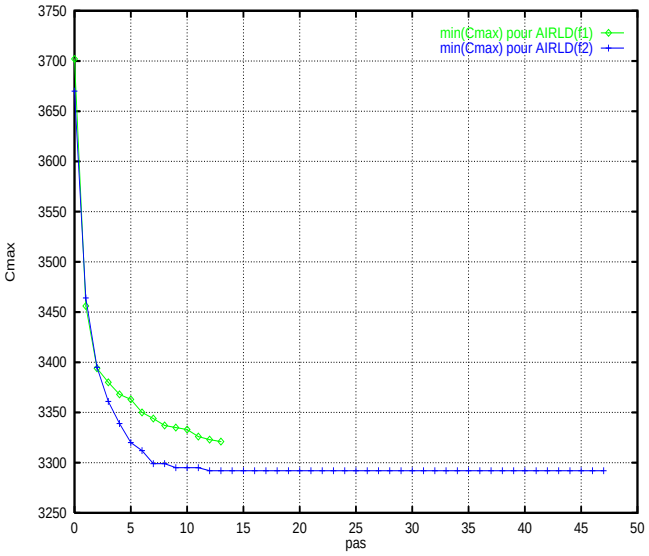


(e) ta01

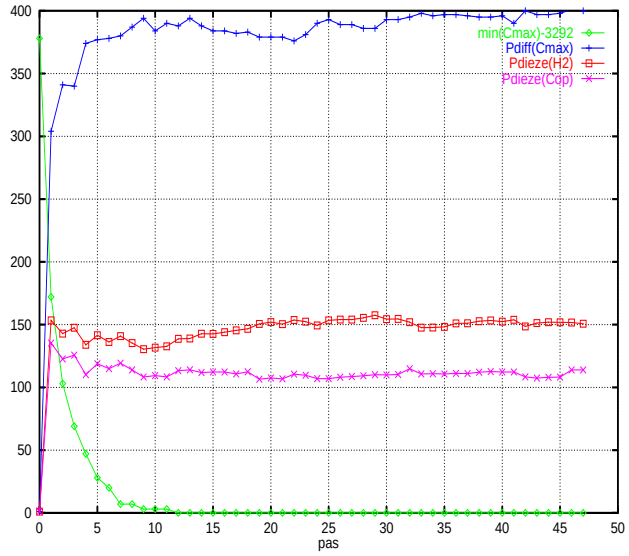


(f) ta31

Figure 7.10: Répartition du makespan des solutions finales pour l'AIRLD. L'axe des abscisses correspond au pourcentage d'écart à l'optimum et l'axe des ordonnées correspond au pourcentage cumulé de solutions finales. Par exemple pour l'instance ft20, les courbes 7.10(b) montrent que le makespan de près de 50% (50.2) des solutions est situé à moins de 16% (15.8) du makespan optimum pour l'AIRLD(f_1) alors que le makespan de près de 50% (50.2) des solutions est situé à moins de 13% (12.8) du makespan optimum pour l'AIRLD(f_6).



(a) Évolutions comparées du makespan durant les plus longues marches obtenues pour AIRLD(f_1) et AIRLD(f_2) sur l'instance ta51.



(b) Évolution de différents paramètres lors de la plus longue marche de l'AIRLD(f_2) sur l'instance ta51.

Figure 7.11: Évolution de $\phi_\sharp$ le long d'une marche de l'AIRLD(f_2) :
(a) est donnée à titre d'exemple pour illustrer la différence de longueur de marche entre AIRLD(f_1) et AIRLD(f_2). La courbe relative à l'AIRL(f_1) correspond à une marche courte (13 pas) interrompue par un plateau de f_1 (critère d'arrêt de l'AIRLD(f_1)). La courbe relative à l'AIRLD(f_2) met en évidence plusieurs plateaux du makespan qui ont été traversés par l'AIRLD pour aboutir à une meilleure solution. Cette courbe se termine par un long plateau (du makespan) sur lequel f_2 prend effectivement des valeurs différentes sans toutefois permettre à l'AIRL d'aboutir à une meilleure solution : la marche est interrompue sur un plateau de f_2 (critère d'arrêt de l'AIRLD(f_2)).
(b) montre l'évolution de différents paramètres au cours de la marche de l'AIRLD(f_2) : la courbe reprend l'évolution du makespan à laquelle nous avons fait subir un décalage d'origine (min(Cmax)-3292) afin de recaler les éventuelles variations des paramètres par rapport à l'évolution du makespan. La courbe (b) montre également les variations de $\phi_\sharp(H_2)$ et $\phi_\sharp(C_{\text{op}})$ (notés Pdieze(H2) et Pdieze(Cop)) ainsi que le nombre de valeurs distinctes du makespan présentes dans le voisinage du point courant (Pdiff(Cmax)). Deux observations découlent de l'étude de (b) : d'une part, aucune variation significative $\phi_\sharp(H_2)$ et $\phi_\sharp(C_{\text{op}})$ ne peut être observée ; d'autre part, la faible valeur de Pdiff(Cmax) confirme la présence de nombreux plateaux dans le voisinage de chaque point le long de la marche. Il est possible d'estimer, de manière très approximative, la taille des plateaux pour f_1 en se basant sur la taille du voisinage soit 275625 voisins (voir tableau 8.9, page 155) et la valeur moyenne de Pdiff(Cmax) le long de la marche soit environ 385.38 : la taille moyenne des plateaux est donnée par la formule $\frac{275625}{385.38} \approx 715$; ce qui représente environ 0.26% de la taille du voisinage. Connaissant la taille moyenne des plateaux, il suffit de déterminer la valeur moyenne de $\phi_\sharp(H_2)$ le long de la marche ($\approx$ 147.49) pour estimer, toujours de manière très approximative, la taille des plateaux pour f_2 : $\frac{275625}{385.38 \times 147.49} \approx 5$. Autrement dit, la taille moyenne estimée des plateaux passe de ≈ 715 pour f_1 à ≈ 5 pour f_2, ce qui illustre parfaitement l'efficacité du critère H_2 dans son rôle de discernement des voisins de même valeur de makespan.

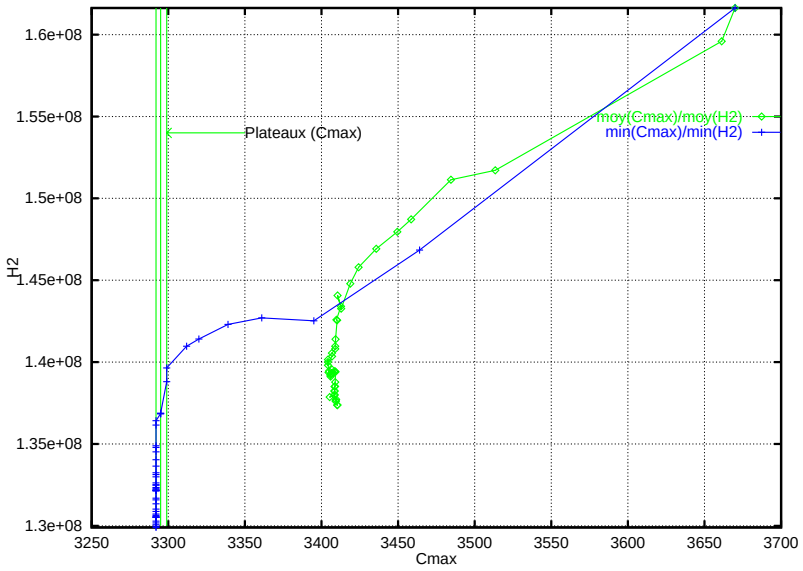


Figure 7.12: Étude des corrélations le long d'une marche de l'AIRLD(f_2) pour l'instance ta51. Pour cette instance, les marches de l'AIRLD(f_2) se terminent quasi-systématiquement à l'issue d'un (long) plateau pour f_1 ce qui se traduit par une corrélation plus faible entre H_2 et le makespan, ainsi sur 10 expériences nous obtenons une corrélation de 0.895 pour les solutions initiales, contre 0.482 pour les solutions finales (optima locaux pour f_2).
La courbe représente l'évolution de l'AIRLD(f_2) dans le plan ($C_{\max}$, H_2) durant la plus longue marche obtenue pour l'instance ta51 : au cours de la marche, l'AIRL évolue de droite à gauche selon les valeurs décroissantes de makespan (axe x), à quelques légères variations près la courbe montre une décroissance simultanée de H_2 (axe y) sauf sur les plateaux du makespan (où la valeur de $C_{\max}$ est invariante par définition).

Nous nous sommes assurés sur un nombre représentatif d'instances et d'expériences que les valeurs de $\phi_\sharp(H_2)$ et $\phi_\sharp(C_{\text{op}})$ étaient suffisamment élevées pour permettre un choix effectif parmi les solutions sur les plateaux. À titre d'exemple, la figure 7.11 présente l'évolution de $\phi_\sharp(H_2)$ et $\phi_\sharp(C_{\text{op}})$ lors de l'évolution d'un AIRLD(f_2). L'exemple choisi correspond à la plus longue marche observée pour l'instance ta51 (50×15) en utilisant soit f_1, soit f_2. Ainsi que cela a été dit lors de l'étude des critères secondaires, obtenir une valeur de $\phi_\sharp$ supérieure à 1 ne suffit pas, il est nécessaire de mesurer la corrélation entre le critère principal et les critères secondaires. Des mesures moyennes ont déjà été effectuées (et présentées dans une section précédente). Nous complétons ces mesures par l'étude d'une évolution de la corrélation entre H_2 et $C_{\max}$ le long d'une marche de l'AIRLD(f_2) (voir figure 7.12).

Le problème majeur de l'AIRLD (utilisé) est lié au calcul complet du voisinage effectué à chaque pas, afin de choisir le meilleur voisin, ce qui entraîne un coût de calcul important. De manière à accélérer la recherche, nous avons défini plusieurs AIRLND qui vont être présentés dans les sections suivantes. Leur principe de fonctionnement diffère de l'AIRLD au niveau de la détermination du prochain voisin à visiter.

7.3.3.2 AIRLND

```
critère-d'arrêt←faux
tentatives-infructueuses←0
Créer une solution initiale s
Répéter ...
   Choisir aléatoirement un voisin s'∈V_O(s)
   Si f(s') < f(s) alors
      s←s'
      tentatives-infructueuses←0
   Sinon
      incrémenter(tentatives-infructueuses)
      Si tentatives-infructueuses≥M²J(J−1) alors
         critère-d'arrêt←vrai
Tant que critère-d'arrêt non satisfait
Afficher("Solution trouvée " s)
```

Figure 7.13: Principe de fonctionnement d'un AIRLND.

La figure 7.13 présente un pseudo-algorithme correspondant au principe de fonctionnement d'un AIRLND. Contrairement à l'AIRLD, cet AIRL ne calcule pas le voisinage de s ($V_{\mathcal{O}}(s)$), mais uniquement une solution s' dans ce voisinage.

Nous avons restreint le test de l'AIRLND aux fonctions coût f_1 à f_4 : les fonctions f_2 ou f_3 sont les fonctions associées aux meilleures performances au cours de tests préliminaires, tandis que la fonction f_4 (bruitée) sert de test de référence. Pour chaque instance, un ensemble de 500 solutions initiales est généré aléatoirement. Ensuite, les fonctions sont testées sur ce même ensemble. Les résultats obtenus sont présentés dans le tableau 7.9.

Pour chaque instance, f_3 obtient les meilleurs résultats en terme de qualité des optima locaux trouvés comme cela était déjà le cas pour l'AIRLD. Les plus mauvais résultats sont obtenus par la fonction initiale f_1 (où $C_{\max}$ est le seul critère). La fonction f_4, qui intègre un générateur de bruit obtient quasi-systématiquement de meilleurs résultats que f_1. Les résultats obtenus par la fonction f_2 sont meilleurs que ceux de f_1 ce qui confirme l'utilité du critère H_2.

Instance	ft10	ft20	ta01	abz7	ta31
Taille	10x10	20x5	15x15	20x15	30x15
Optimum	930	1165	1231	656	1764
f_1 Moyenne	1084	1353	1460	782	2150
Écart-type	40	43	47	19	46
Minimum	991 (6.6)	1232 (5.8)	1350 (9.7)	738 (12.5)	2020 (14.4)
Maximum	1233	1509	1600	853	2310
f_2 Moyenne	1069	1332	1442	775	2128
Écart-type	39	46	46	19	44
Minimum	962 (3.4)	1220 (4.7)	1334 (8.4)	732 (11.6)	2018 (14.3)
Maximum	1211	1473	1606	836	2271
f_3 Moyenne	1068	1329	1440	774	2125
Écart-type	39	47	46	19	45
Minimum	962 (3.4)	1208 (3.7)	1334 (8.4)	725 (10.5)	1961 (11.1)
Maximum	1211	1470	1606	836	2267
f_4 Moyenne	1076	1338	1446	777	2139
Écart-type	39	46	42	17	47
Minimum	970 (4.3)	1216 (4.4)	1343 (9.1)	731 (11.4)	2033 (15.2)
Maximum	1233	1513	1573	843	2312

Tableau 7.9: Résultats de l'AIRLND.

Comme pour l'AIRLD, nous pouvons établir un classement entre les différentes fonctions coût selon les performances :

$$f_3 > f_2 > f_4 > f_1$$

où $f_3 > f_2$ signifie que f_3 donne de meilleurs résultats par rapport à f_2.

Le classement est similaire en considérant les performances moyennes ou la répartition des valeurs des solutions finales (voir table 7.14).

Si l'on considère les performances moyennes, le gain sur les cinq instances considérées est de l'ordre de 1.61% entre AIRLND(f_1) et AIRLND(f_3). Le gain, plus faible que pour l'AIRLD, peut s'expliquer par le caractère non-déterministe de l'AIRLND, mais aussi par le nombre d'appels à la fonction coût qui est bien moins important que pour l'AIRLD comme le montre le paragraphe suivant.

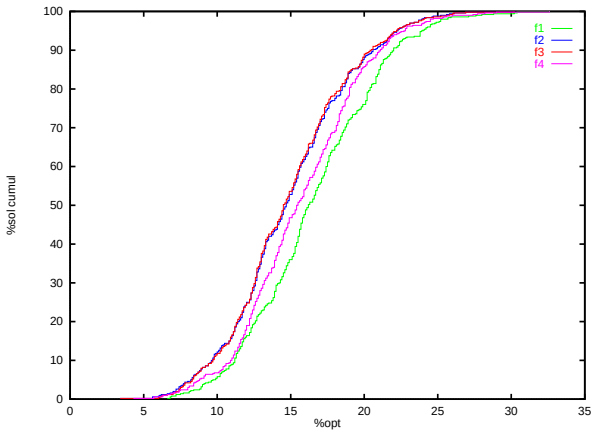
Les tableaux 8.10 et 8.11 (page 155) permettent de comparer précisément le nombre d'évaluations pour les deux AIRL. En ce point nous donnons un ordre de grandeur moyen sur le nombre d'évaluations pour les instances considérées : l'AIRLND(f_1) nécessite 2.42 fois moins d'évaluations que l'AIRLD(f_1), l'AIRLND(f_2) nécessite 5.30 fois moins d'évaluations que l'AIRLD(f_2), et l'AIRLND(f_3) nécessite 5.48 fois moins d'évaluations que l'AIRLD(f_3).

En conclusion de cette section, l'AIRLND confirme l'utilité des critères secondaires, bien que le gain relatif entre f_3 et f_1 soit plus faible. Par rapport à l'AIRLD, les résultats moyens sont comparables alors que les meilleurs résultats (minimum) sont obtenus par l'AIRLND tout en nécessitant beaucoup moins d'appels à la fonction coût.

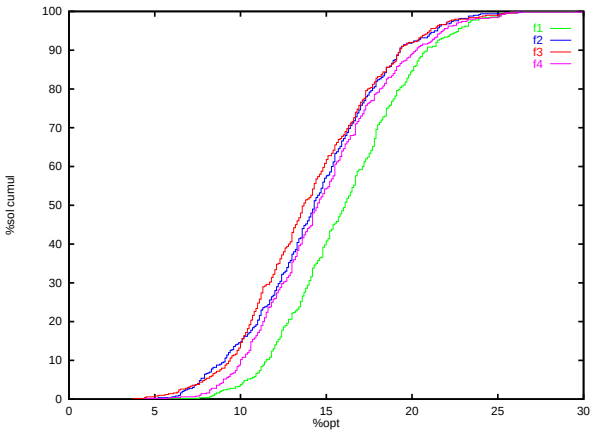
Afin d'améliorer les performances de l'AIRLND, nous le dotons d'une mémoire pour obtenir l'AIRLNDM présenté en section suivante.

7.3.3.3 AIRLNDM

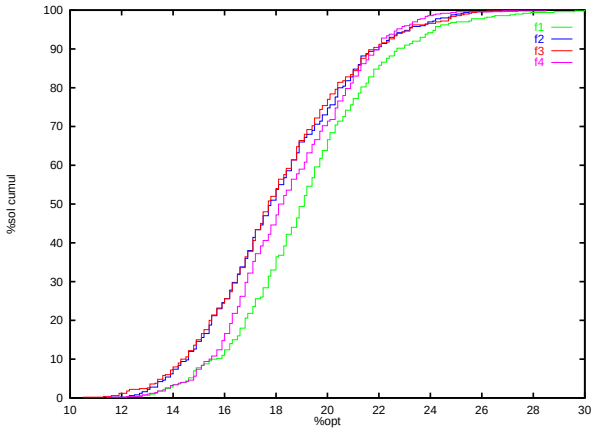
Un AIRLNDM est un algorithme itératif de recherche locale, non déterministe, utilisant une mémoire de mouvements.



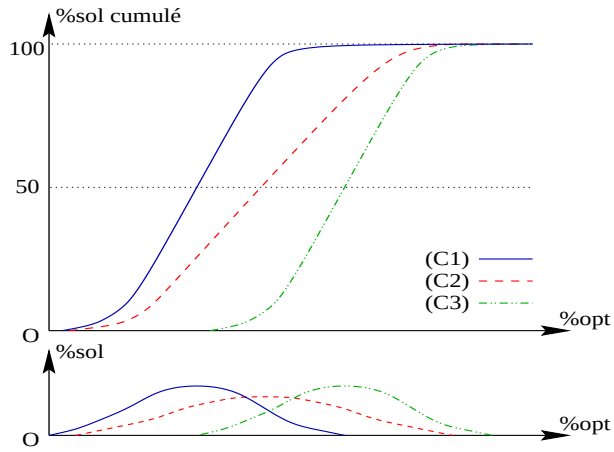
(a) ft10



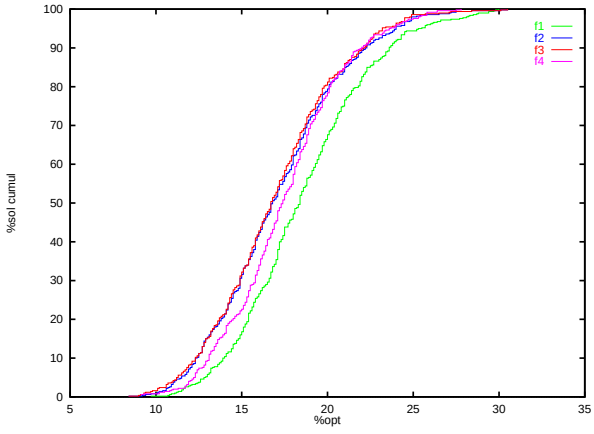
(b) ft20



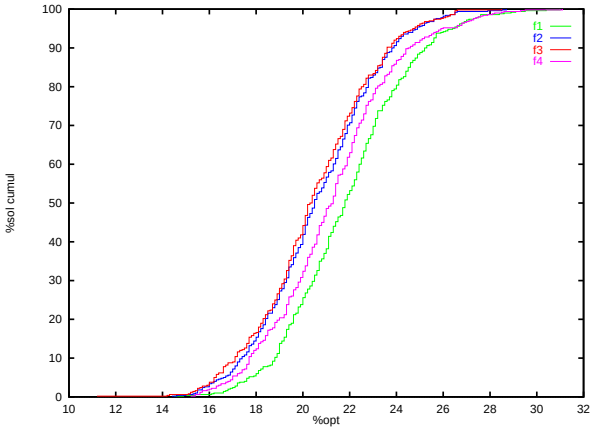
(c) abz7



(d) Relation entre % et % cumulé



(e) ta01



(f) ta31

Figure 7.14: Répartition du makespan des solutions finales pour l'AIRLND. L'axe des abscisses correspond au pourcentage d'écart à l'optimum et l'axe des ordonnées correspond au pourcentage cumulé de solutions finales.

```
critère-d'arrêt←faux
tentatives-infructueuses←0
mémoire-mouvements←∅
Créer une solution initiale s
Répéter ...
   Choisir aléatoirement un voisin s'∈V_O(s)
                          tel que s'∉mémoire-mouvements
   Si f(s') < f(s) alors
      s←s'
      tentatives-infructueuses←0
      mémoire-mouvements←∅
   Sinon
      incrémenter(tentatives-infructueuses)
      insérer mouvement(s→s') dans mémoire-mouvements
      Si tentatives-infructueuses≥M²J(J−1)/2 alors
         critère-d'arrêt←vrai
Tant que critère-d'arrêt non satisfait
Afficher("Solution trouvée " s)
```

Figure 7.15: Principe de fonctionnement d'un AIRLNDM.

Instance	ft10	ft20	ta01	abz7	ta31
Taille	10x10	20x5	15x15	20x15	30x15
Optimum	930	1165	1231	656	1764
f_1 Moyenne	1084	1349	1460	781	2150
Écart-type	42	43	47	19	47
Minimum	991 (6.6)	1239 (6.4)	1351 (9.7)	738 (12.5)	2020 (14.4)
Maximum	1233	1509	1600	853	2339
f_2 Moyenne	1068	1330	1442	774	2129
Écart-type	39	48	45	19	44
Minimum	955 (2.7)	1211 (3.9)	1334 (8.4)	732 (11.6)	2020 (14.4)
Maximum	1180	1473	1606	836	2271
f_3 Moyenne	1067	1325	1440	774	2126
Écart-type	38	46	44	19	43
Minimum	951 (2.3)	1215 (4.3)	1334 (8.4)	732 (11.6)	2020 (14.4)
Maximum	1180	1470	1606	836	2267

Tableau 7.10: Résultats de l'AIRLNDM.

La figure 7.15 présente un pseudo-algorithme correspondant au principe de fonctionnement d'un AIRLNDM.

L'AIRLNDM a été testé en utilisant les fonctions coût f_1 à f_3. Pour chaque instance, un ensemble de 500 solutions initiales est généré aléatoirement. Ensuite, les fonctions sont testées sur ce même ensemble. Les résultats obtenus sont présentés dans le tableau 7.10.

En moyenne, f_3 obtient les meilleurs résultats en terme de qualité des optima locaux trouvés comme cela était déjà le cas pour l'AIRLND. Les plus mauvais résultats sont obtenus par la fonction initiale f_1. Les résultats obtenus par la fonction f_2 sont meilleurs que ceux de f_1, ce qui confirme l'utilité du critère H_2. Le gain entre f_2 et f_3 est plus faible que pour l'AIRLND : comme pour la comparaison entre l'AIRLD et l'AIRLND, l'explication vient de la forte réduction du nombre d'appels à la fonction coût (il y a un facteur moyen ≈ 1.6 entre l'AIRLND et l'AIRLNDM). Pour f_3, la dispersion (en terme d'écart-type) des solutions, inférieure ou égale à celle de f_2, signifie que f_3 est plus *robuste* que f_2 : pour f_3, les solutions sont davantage groupées autour de la moyenne, ce qui confirme les meilleurs résultats en moyenne de f_3 par rapport à f_2.

Comme pour l'AIRLND, nous pouvons établir un classement entre les différentes fonctions coût selon les performances (voir table 7.16) :

$$f_3 \geq f_2 > f_1$$

où $f_3 > f_2$ signifie que f_3 donne de meilleurs résultats par rapport à f_2.

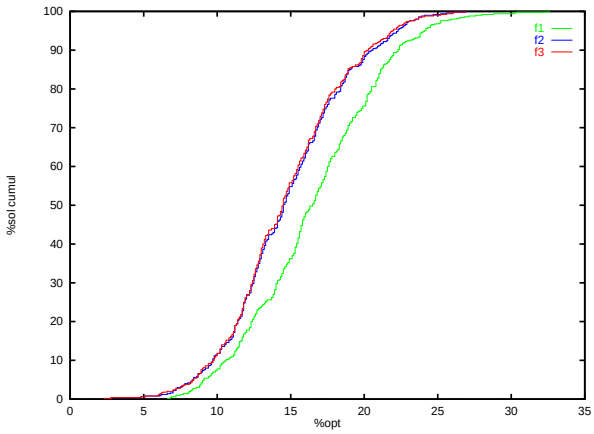
Si l'on considère les performances moyennes, le gain sur les cinq instances considérées est de l'ordre de 1.59% entre AIRLNDM(f_1) et AIRLNDM(f_3). Le gain, plus faible que pour l'AIRLND, peut s'expliquer par le nombre d'appels à la fonction coût qui est moins important que pour l'AIRLD et l'AIRLND, comme le montre le paragraphe suivant.

Les tableaux 8.10 et 8.11 (page 155) permettent de comparer précisément le nombre d'évaluations pour les deux AIRL. En ce point nous donnons un ordre de grandeur moyen sur le nombre d'évaluations : l'AIRLNDM(f_1) nécessite 4.15 fois moins d'évaluations que l'AIRLD(f_1), l'AIRLNDM(f_2) nécessite 8.82 fois moins d'évaluations que l'AIRLD(f_2), et l'AIRLNDM(f_3) nécessite 9.13 fois moins d'évaluations que l'AIRLD(f_3).

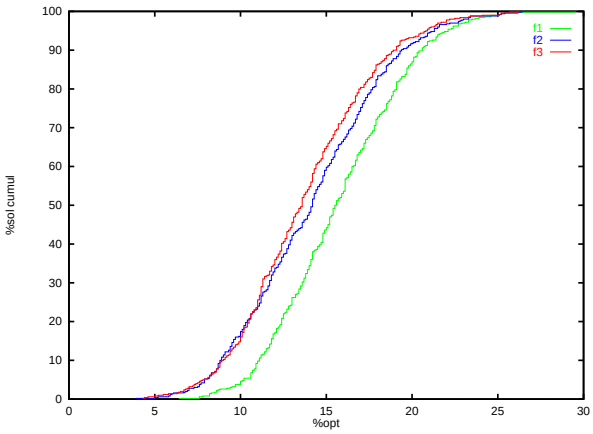
Nous effectuons la même comparaison entre l'AIRLND et l'AIRLNDM pour illustrer l'impact de la mémoire sur le nombre d'évaluations : l'AIRLNDM(f_1) nécessite 1.48 fois moins d'évaluations que l'AIRLND(f_1), l'AIRLNDM(f_2) nécessite 1.66 fois moins d'évaluations que l'AIRLND(f_2), et l'AIRLNDM(f_3) nécessite 1.64 fois moins d'évaluations que l'AIRLND(f_3).

L'ajout de la mémoire a surtout eu un impact sur le nombre d'appels à la fonction coût (et par conséquent sur la durée d'exécution). L'impact sur la qualité des solutions finales est beaucoup moins significatif. À ce propos, le cas de l'instance ft20 est intéressant, car l'AIRLNDM(f_2) aboutit à un meilleur résultat (minimum) que l'AIRLNDM(f_3) bien qu'en moyenne l'AIRLNDM(f_3) donnent de meilleurs résultats. Ceci peut être dû à un effet de bord de f_3 ou à un effet dû *au hasard* (lors de l'échantillonnage des solutions initiales, ...). Pour clarifier ce problème, nous disposons d'un grand nombre d'expériences lancées pour des tests préliminaires ou pour des mesures présentées dans le cadre de notre étude. Nous utilisons les résultats présentés dans le tableau A.5 (page 176). Parmi une centaine d'instances, nous avons obtenu dans près de 18% des instances de meilleurs résultats[22] pour AIRLNDM(f_2) par rapport AIRLNDM(f_3). De même, dans environ 18% des instances, de meilleurs résultats pour AIRLNDM(f_2) par rapport AIRLNDM(f_3) en comparant du point de vue de la moyenne du makespan des solutions finales. Comme cela a été dit précédemment, la comparaison en terme de makespan minimum peut être altérée par un effet de bord lié au caractère

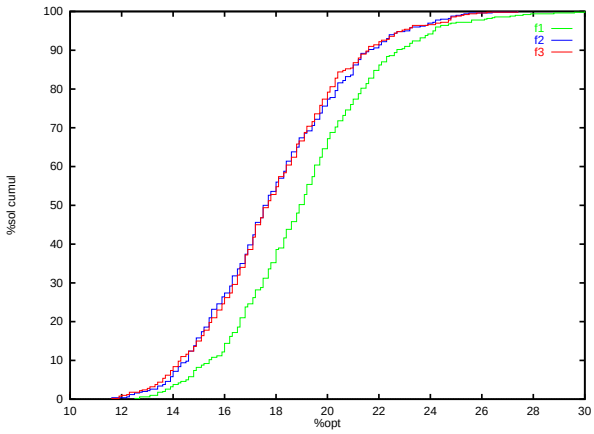
[22]La comparaison est établie sur la plus petite valeur de makespan obtenue parmi l'ensemble des solutions finales pour chacun des deux AIRLNDM.



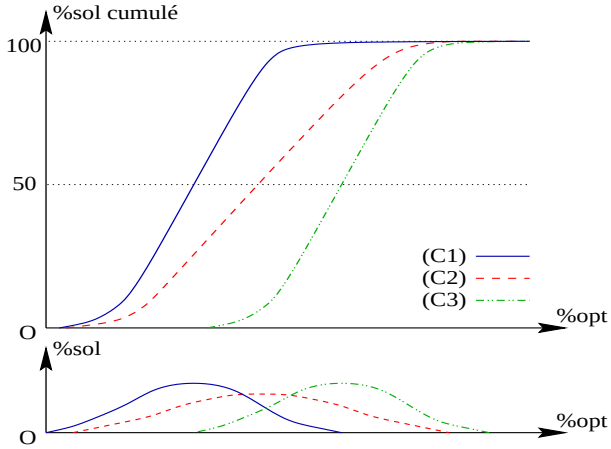
(a) ft10



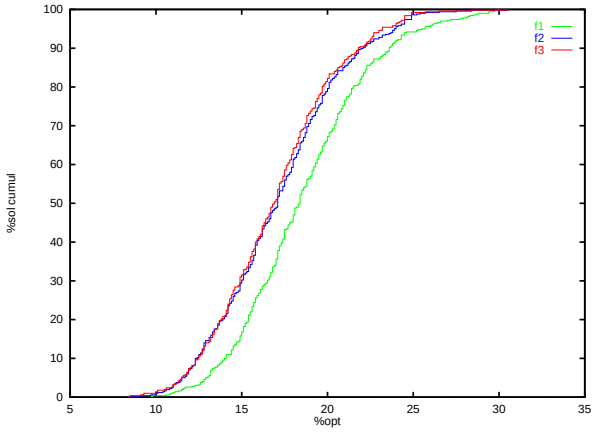
(b) ft20



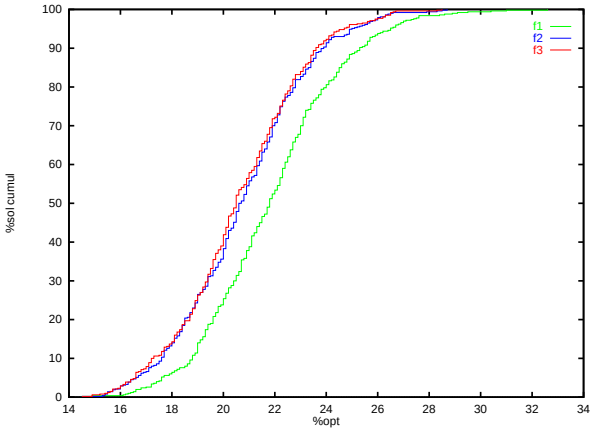
(c) abz7



(d) Relation entre % et % cumulé



(e) ta01



(f) ta31

Figure 7.16: Répartition du makespan des solutions finales pour l'AIRLNDM. L'axe des abscisses correspond au pourcentage d'écart à l'optimum et l'axe des ordonnées correspond au pourcentage cumulé de solutions finales.

stochastique de l'AIRLNDM. Par contre, les résultats moyens sont plus représentatifs. Ces résultats sont confortés par le fait que dans près de 15% des cas, le meilleur résultat de f_2 par rapport f_3 est accompagné d'une réduction de l'écart-type, traduisant un recentrage des solutions finales autour de la moyenne. Autrement dit, pour environ 18% des instances, il est plus intéressant d'utiliser f_2 que f_3 au sein de l'AIRLNDM.

Pour tenter d'expliquer le comportement des AIRLNDM pour ces instances où f_2 donne de meilleurs résultats, considérons un point x dans l'espace de recherche choisi de manière à ce que le voisinage comporte un plateau p dont les points correspondent à des solutions de coût minimum pour le voisinage de x en considérant $f_2 : C(p) = \min_{x' \in V_{\mathcal{O}}(x)} C(x')$ avec $C(x) = f_2(x)$ et $p \subset V_{\mathcal{O}}(x)$. Si nous plaçons AIRLNDM(f_2) et AIRLNDM(f_3) sur x, l'AIRLNDM(f_2) choisit au hasard le prochain point visité parmi plusieurs solutions voisines de même coût appartenant à p. Par contre l'AIRLNDM(f_3) placé sur ce même point x verra soit un plateau de taille inférieure à la taille de p si f_3 ne permet pas de discerner les points de p du point de vue du coût, soit un ensemble de points de coûts différents. Dans les deux cas AIRLNDM(f_3) a de très fortes chances de ne pas « voir » le même plateau p car f_3 comporte un critère secondaire en plus par rapport à f_2. Donc localement, f_3 permet de mieux discerner les points du voisinage et par conséquent d'effectuer un meilleur choix. En revanche en itérant ce processus, la succession des choix effectués par l'AIRLNDM(f_3) le conduit vers une solution moins intéressante que la solution de l'AIRLNDM(f_2). Donc, f_3 permet d'effectuer localement un meilleur choix parmi les voisin de x, mais à long terme (*i.e.* après plusieurs pas) le rôle « d'oracle » joué par les critères secondaires de f_3 est néfaste à l'AIRLNDM(f_3). Il serait possible de mettre en œuvre une méthode poussée d'analyse des résultats, mais il semble raisonnable de penser que la faible corrélation entre le makespan et le nombre d'opérations critiques (critère différenciant f_3 de f_2) ainsi que la décroissance du nombre d'appels à la fonction coût ont eu raison du faible gain apporté par f_3 par rapport à f_2.

Dans la majorité des cas f_3 améliore les résultats. C'est pourquoi nous poursuivons l'étude en considérant f_3 tout en gardant en mémoire les remarques précédentes. Malgré la forte réduction du nombre d'appels à la fonction coût, f_2 apporte encore un gain significatif pour l'AIRLNDM.

Afin d'améliorer les performances de l'AIRLNDM, nous étendons le mécanisme de mémorisation de manière à tenir compte des déplacements sur les plateaux pour obtenir l'AIRLNDPM présenté en section suivante.

7.3.3.4 AIRLNDPM

Un AIRLNDPM est un algorithme itératif de recherche locale, non déterministe, utilisant une mémoire de mouvements et capable de se déplacer sur les plateaux.

La figure 7.17 présente un pseudo-algorithme correspondant au principe de fonctionnement d'un AIRLNDPM.

L'AIRLNDPM a été testé en utilisant les fonctions coût f_1 à f_3. Pour chaque instance, les fonctions sont testées sur un ensemble commun de 500 solutions initiales. Les résultats obtenus sont présentés dans le tableau 7.11

Si l'on considère le coût minimum obtenu pour les différentes fonctions, f_2 donne de meilleurs résultats que f_3. Par contre, pour le coût moyen, f_3 obtient les meilleurs résultats comme cela était déjà le cas pour l'AIRLNDM. Les plus mauvais résultats sont obtenus par la fonction f_2. Pour f_3, la dispersion (en terme d'écart-type) des solutions est inférieure ou égale à celle de f_2, ce qui signifie que f_3 est meilleure que f_2 en terme de *robustesse* : pour f_3, les solutions sont davantage groupées autour de la moyenne, ce qui confirme les meilleurs résultats en moyenne de f_3 par rapport à f_2.

Contrairement aux résultats obtenus pour les précédents AIRL, les résultats obtenus par la

```
critère-d'arrêt←faux
tentatives-infructueuses←0
pas-sur-plateau←0
mémoire-mouvements←∅
Créer une solution initiale s
Répéter ...
   Choisir aléatoirement un voisin s'∈V_O(s)
                          tel que s'∉mémoire-mouvements
   Si f(s')≤f(s) alors
      Si f(s') = f(s) alors
         incrémenter(pas-sur-plateau)
         Si pas-sur-plateau≥M²J(J−1) alors
            critère-d'arrêt←vrai
      Sinon
         pas-sur-plateau←0
      s←s'
      tentatives-infructueuses←0
      mémoire-mouvements←∅
   Sinon
      incrémenter(tentatives-infructueuses)
      insérer mouvement(s→s') dans mémoire-mouvements
      pas-sur-plateau←0
      Si tentatives-infructueuses≥M²J(J−1)/2 alors
         critère-d'arrêt←vrai
Tant que critère-d'arrêt non satisfait
Afficher("Solution trouvée " s)
```

Figure 7.17: Principe de fonctionnement d'un AIRLNDPM.

Instance	ft10	ft20	ta01	abz7	ta31
Taille	10x10	20x5	15x15	20x15	30x15
Optimum	930	1165	1231	656	1764
f_1 Moyenne	987	1206	1283	693	1843
Écart-type	24	24	19	9	29
Minimum	937 (0.75)	1174 (0.77)	1241 (0.81)	673 (2.6)	1781 (0.91)
Maximum	1066	1307	1349	726	1945
f_2 Moyenne	993	1201	1307	699	1882
Écart-type	25	23	24	11	40
Minimum	937 (0.75)	1165 (0.0)	1248 (1.4)	674 (2.7)	1786 (1.2)
Maximum	1079	1308	1404	740	2013
f_3 Moyenne	993	1200	1306	699	1880
Écart-type	25	23	23	11	37
Minimum	937 (0.75)	1165 (0.0)	1254 (1.9)	675 (2.9)	1785 (1.2)
Maximum	1078	1290	1424	746	2003

Tableau 7.11: Résultats de l'AIRLNDPM obtenus à partir de 500 exécutions pour chaque paramétrage (instance, fonction coût).

fonction f_1 sont meilleurs que ceux de f_2 ce qui semble remettre en cause l'utilité des critères secondaires. Comme pour l'AIRLNDM, le gain entre f_2 et f_3 est faible.

Comme pour les AIRL étudiés dans les sections précédentes, nous pouvons établir un classement entre les différentes fonctions coût selon les performances. Excepté pour ft20 (où $f_3 > f_2 > f_1$), le classement suivant reflète les performances relatives des trois fonctions coût étudiées :

$$f_1 > f_3 \geq f_2$$

où $f_1 > f_3$ signifie que f_1 donne de meilleurs résultats par rapport à f_3.

Ces résultats sont confirmés par l'étude des répartitions des valeurs de makespan des solutions finales de l'AIRLNDM (voir table 7.18).

Pour tenter d'expliquer ce résultat, l'hypothèse avancée pour les AIRLND précédents ne tient plus : la dégradation des performances (entre f_1 et les fonctions multicritères) ne peut pas s'expliquer par le nombre d'appels à la fonction coût car il est plus important que pour l'AIRLD, l'AIRLND et l'AIRLNDM.

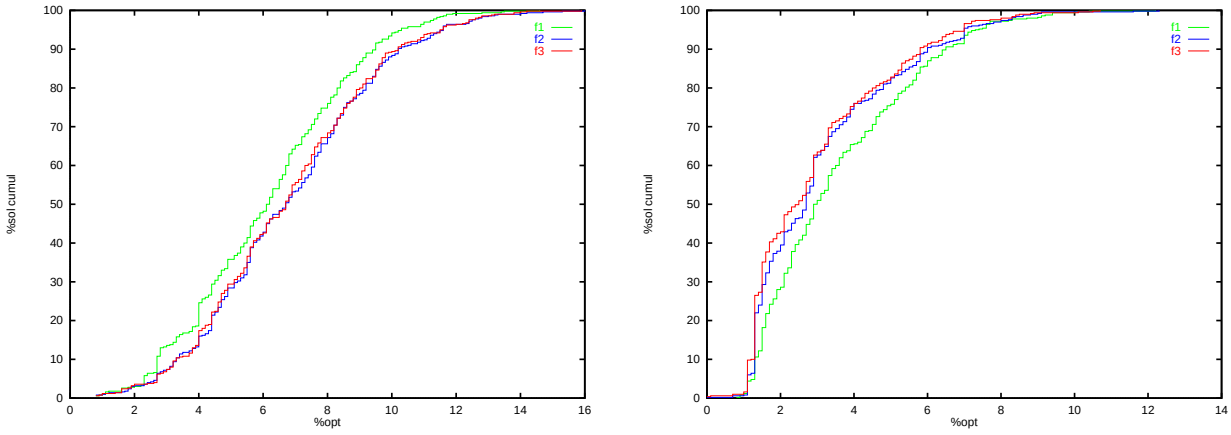
En contrepartie, AIRLNDPM(f_1) et AIRLNDPM(f_2) obtiennent les meilleurs résultats parmi les AIRL : les performances obtenues sont assez remarquables car les meilleurs résultats se situent à 2.6% pour abz7, à moins de 1% pour (ft10, ta01, ta31), et la solution optimale est obtenue pour ft20. Le tableau 7.12 montre que l'optimum est découvert une fois sur 500 pour AIRLNDPM(f_2), et deux fois sur 500 AIRLNDPM(f_3). Bien que ces deux fréquences soient relativement faibles, les 10 premières valeurs des fréquences cumulées[23] relatives à l'histogramme de répartition des makespan des solutions indiquent que l'AIRLNDPM(f_3) est plus performant que l'AIRLNDPM(f_1) et l'AIRLNDPM(f_3). Ces résultats sont confirmés par les courbes de répartition des valeur de makespan des solutions finales (voir courbes 7.18(b) et 7.18(d)).

f_1			f_2			f_3		
val	fréq	cum	val	fréq	cum	val	fréq	cum
1174	1	1	1165	1	1	1165	2	2
1175	2	3	1173	2	3	1166	1	3
1176	1	4	1177	1	4	1173	2	5
1177	2	6	1178	26	30	1177	3	8
1178	16	22	1179	2	32	1178	41	49
1179	2	24	1180	78	110	1179	1	50
1180	29	53	1181	10	120	1180	82	132
1181	8	61	1182	21	141	1181	4	136
1182	20	81	1183	5	146	1182	34	170
1183	10	91	1184	15	161	1183	5	175

Tableau 7.12: Ce tableau présente les fréquences (notées fréq) des dix meilleures valeurs de makespan pour l'AIRLNDPM(f_1), l'AIRLNDPM(f_2) et l'AIRLNDPM(f_3), sur un ensemble de 500 solutions finales pour chaque fonction. L'instance utilisée est ft20, le makespan optimum est 1165. Les fréquences cumulées (cum) permettent de comparer le nombre de solutions finales dont le makespan est inférieur ou égal à un seuil fixé. Par exemple, sur 500 solutions finales AIRLNDPM(f_1) engendre 91 solutions de makespan$\leq$ 1183, AIRLNDPM(f_2) engendre 146 solutions de makespan$\leq$ 1183, et AIRLNDPM(f_3) engendre 175 solutions de makespan$\leq$ 1183.

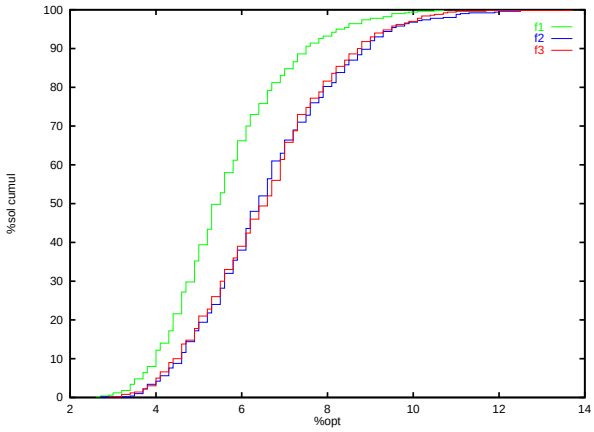
Tels qu'ils sont présentés dans le tableau 7.11, les résultats de l'AIRLNDPM montrent que l'intégration des critères secondaires détériore les résultats. Nous pouvons envisager une explication de ces résultats en nous basant sur le fait que l'AIRLNDPM(f_1) a la possibilité de se déplacer sur des

[23]Les fréquences calculées sur les 500 solutions finales sont utilisées pour réaliser la courbe 7.18(d). Les fréquences cumulées calculées sur les 500 solutions finales sont utilisées pour réaliser la courbe 7.18(b).

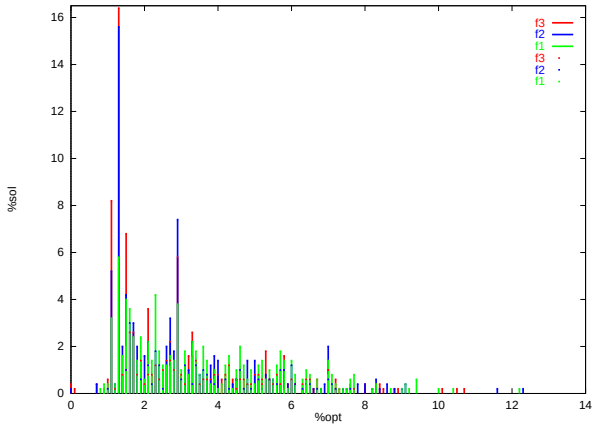


(a) ft10

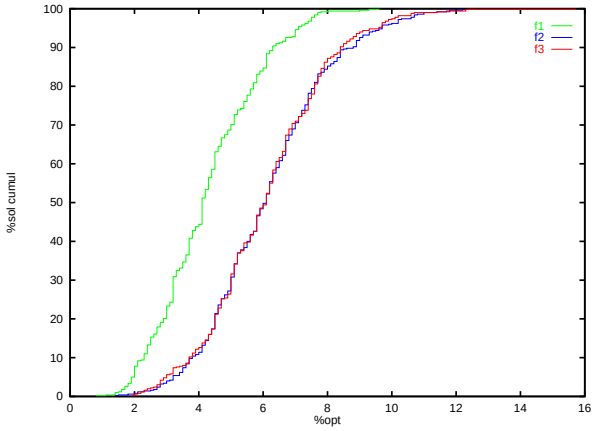
(b) ft20



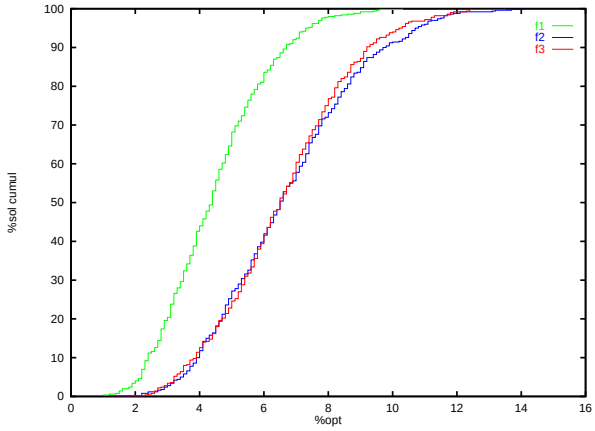
(c) abz7



(d) ft20 : % non cumulé



(e) ta01



(f) ta31

Figure 7.18: Répartition du makespan des solutions finales pour l'AIRLNDPM. L'axe des abscisses correspond au pourcentage d'écart à l'optimum et l'axe des ordonnées correspond au pourcentage cumulé de solutions finales (sauf pour les histogrammes 7.18(d)).

plateaux de plus grande taille que l'AIRLNDPM(f_3) en raison de la modification du paysage induite par l'ajout des critères. De ce fait, AIRLNDPM(f_1) explore des fractions plus faibles des plateaux par rapport à AIRLNDPM(f_2) ou AIRLNDPM(f_3), ce qui lui permet de s'échapper plus facilement des optima locaux alors que les autres AIRLNDPM exploitent de manière plus systématique les plateaux avec pour conséquence le risque élevé de découverte d'un optimum local.

Enfin, la comparaison directe entre l'AIRLNDPM et les autres AIRL doit être effectuée avec beaucoup de précautions car seul l'AIRLNDPM est capable de se déplacer sur les plateaux. Ces évolutions permettent, certes, d'obtenir une solution finale de meilleure qualité, mais au prix de nombreux appels à la fonction coût (évaluations). C'est pourquoi nous avons comparé les résultats obtenus par différents AIRLNDPM au bout d'un nombre fixe d'évaluations (voir figure 7.19).

La comparaison peut être effectuée en deux étapes (pour un même nombre d'appels à la fonction coût) : dans un premier temps, nous comparons AIRLNDPM(f_1) à AIRLNDM(f_2) ou AIRLNDM(f_3) afin de déterminer si l'ajout de critères secondaires dans un AIRL non autorisé à se déplacer sur les plateaux permet d'obtenir de meilleures performances par rapport à un AIRLNDPM n'utilisant pas les critères secondaires (figure 7.19). Dans un second temps, nous comparons les performances de l'AIRLNDM(f_3) à celles de l'AIRLNDPM(f_1) afin de déterminer si l'AIRLNDM(f_3) (non autorisé à se déplacer sur les plateaux), est capable de rivaliser avec l'AIRLNDPM(f_1). L'AIRLNDM utilise une fonction coût multicritère afin d'évaluer comparativement l'impact de cette fonction sur un AIRLNDM par rapport au déplacement sur les plateaux d'un AIRLNDPM(f_1) (voir figure 7.20).

Les courbes 7.19(c) confirment indirectement la réduction de la taille des plateaux suite à l'utilisation de fonctions multicritères, par l'observation de la réduction du nombre de pas (en moyenne) effectués sur les plateaux. Ce résultat se vérifie sur toutes les instances étudiées, y-compris ft20 qui obtient des résultats atypiques (en terme de répartition de makespan de solution finales) par rapport aux autres instances.

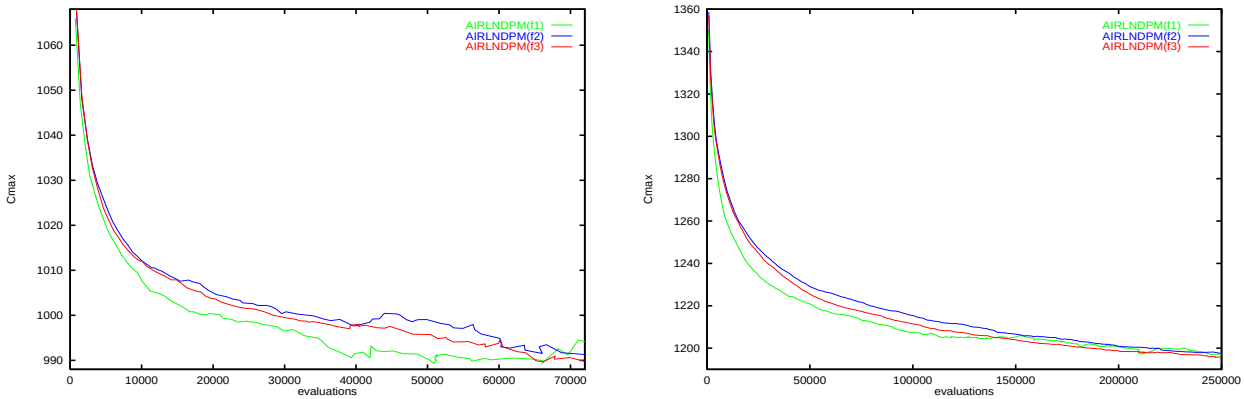
Avant de commenter les résultats obtenus, il semble judicieux de rappeler l'objectif de cette section : il s'agit de déterminer si l'ajout de critères secondaires permet de mieux guider la méthode de recherche dans le paysage.

En utilisant un AIRLNDPM, nous avons introduit un mécanisme non déterministe qui va à l'encontre de l'utilisation des fonctions multicritères : l'objectif de ces fonctions est de réduire la taille des plateaux au voisinage du point courant. Hors des plateaux des fonctions multicritères, l'AIRL est correctement guidé. Sur les plateaux des fonctions multicritères, l'AIRLNDPM est guidé vers des zones peu intéressantes du point de vue du makespan. Autrement dit pour l'AIRLNDPM, tout se passe comme si, pour f_2 ou f_3, nous avions « enlevé des plateaux de f_1 les fractions de plateaux qui conduisaient à de meilleures solutions ».

En résumé les fonctions multicritères sont efficaces tant que l'on ne cherche pas à se déplacer sur « leurs plateaux », sinon tout se passe comme si ces plateaux conduisaient l'AIRLNDPM hors des zones contenant des optima locaux de bonne qualité. Le problème vient du fait que, d'une part, nous cherchons à réduire la taille des plateaux de f_1 et d'autre part, nous cherchons à nous déplacer sur les plateaux des fonctions multicritères ($f \neq f_1$).

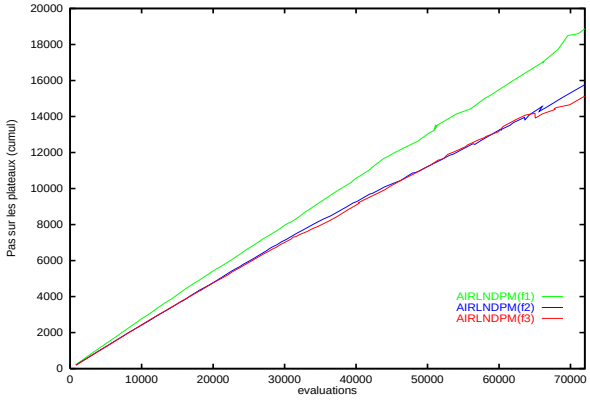
L'ajout du mécanisme de déplacement sur les plateaux a surtout eu un impact sur le nombre d'appels à la fonction coût (et par conséquent sur la durée d'exécution). Au cours de l'étude de l'impact sur la qualité des solutions, nous avons volontairement mis en évidence le problème lié aux comparaisons directes entre méthodes de recherche[24].

[24]Nous montrons au passage que le problème de comparaison entre méthodes de recherche se pose déjà pour de simples AIRL.

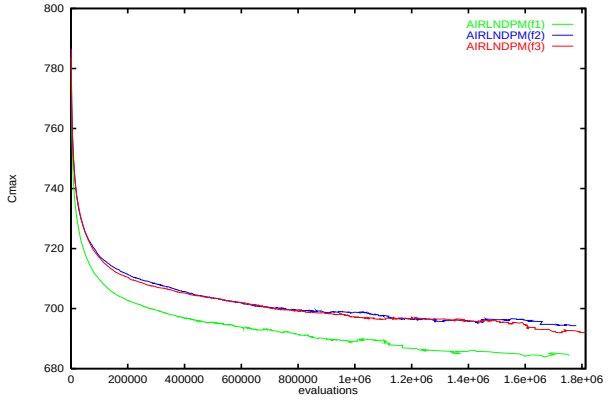


(a) ft10

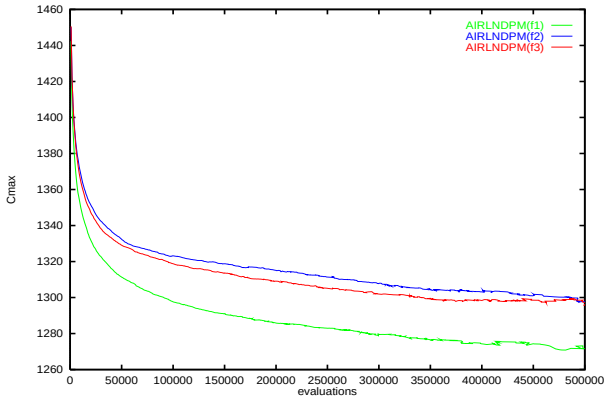
(b) ft20



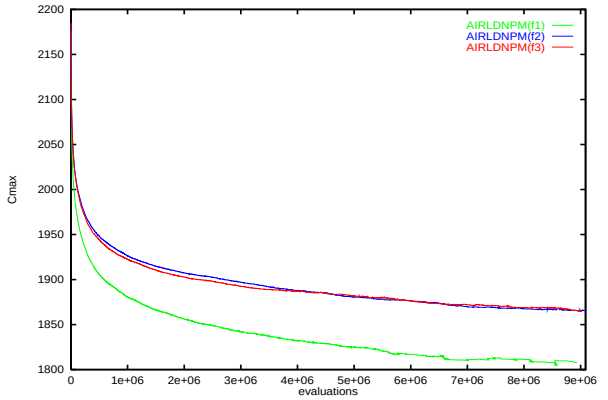
(c) ft10 – cumul des pas



(d) abz7



(e) ta01



(f) ta31

Figure 7.19: Évolutions de l'AIRLNDPM en fonction du nombre d'évaluations. L'axe des abscisses correspond au nombre d'évaluations, l'axe des ordonnées à la valeur moyenne du makespan (sauf pour les courbes 7.19(c)). La moyenne est calculée par pas de 1000 évaluations. À chaque pas, le nombre total d'évaluations e est utilisé comme un seuil : seules les n expériences comportant au moins e évaluations interviennent dans le calcul de la moyenne (initialement $n = 500$ expériences par fonction). n décroît à mesure que e augmente, jusqu'à engendrer des instabilités dans le calcul de la moyenne. Aussi, nous avons tronqué les courbes selon les abscisses lorsque ces instabilités altèrent la lecture des courbes.
Les courbes 7.19(c) montrent qu'il y a effectivement une réduction du nombre de pas effectués sur les plateaux lorsque les fonctions multicritères sont utilisées.

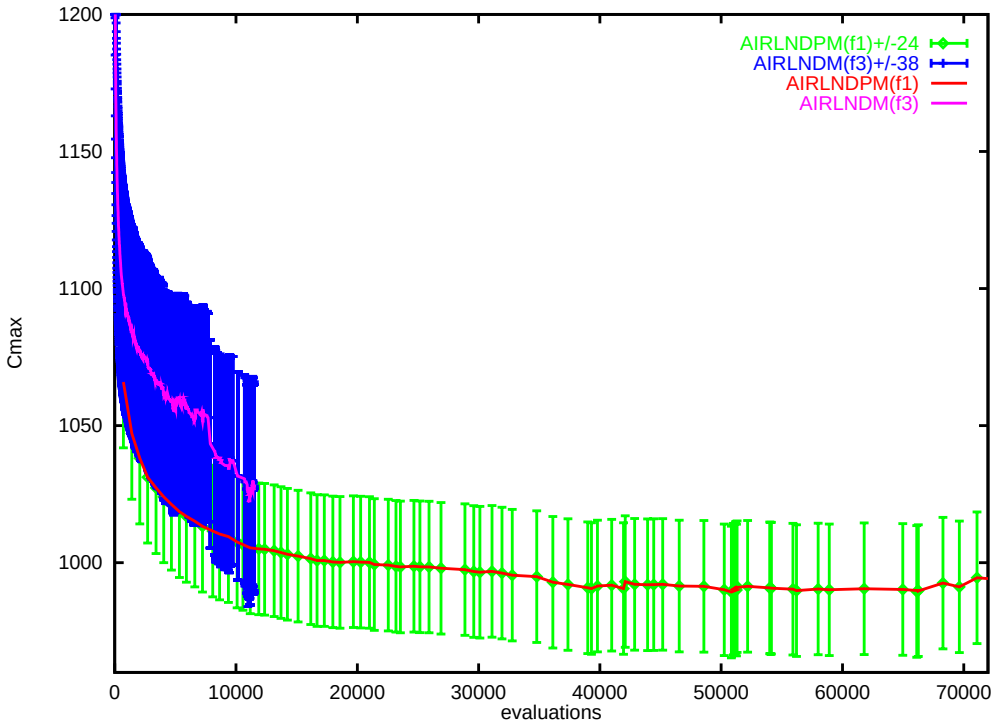


Figure 7.20: Comparaison des évolutions de l'AIRLNDPM(f_1) et de l'AIRLNDM(f_3) pour l'instance ft10. Le principe du calcul de l'évolution moyenne des AIRL est similaire au calcul utilisé pour réaliser la figure 7.19. Étant donné que les marches de l'AIRLNDM sont courtes, les instabilités liées au calcul de la moyenne apparaissent très tôt lors du calcul relatif à l'évolution moyenne de l'AIRLNDM(f_3). Ceci explique les variations selon l'axe des ordonnées pour la courbe relative à l'AIRLNDM. La comparaison directe des évolutions moyennes de l'AIRLNDM(f_3) et de l'AIRLNDPM(f_1) est « trompeuse ». En effet, l'AIRLNDM(f_3) semble donner de moins bons résultats que l'AIRLNDPM(f_1) tout au long des marches. Cependant, les résultats sont différents si l'on s'intéresse à la répartition des résultats le long des marches : considérons la valeur de l'écart-type des solutions finales comme ordre de grandeur pour la répartition des solutions (en prenant pour hypothèse que les variations de l'écart-type soient faibles le long de l'ensemble des 500 marches « recalées par rapport au nombre d'évaluations ») pour réaliser les courbes AIRLNDM(f_3)±38 et AIRLNDPM(f_1)±24. Dans ces conditions, les courbes susnommées représentent l'évolution de près de 70% des marches. Pour l'instance ft10, l'écart-type de l'AIRLNDM(f_3) vaut 38 contre 24 pour l'AIRLNDPM(f_1). En moyenne, il est environ deux fois (≈ 1.89) plus élevé sur l'ensemble des instances étudiées lorsque l'on compare l'AIRLNDM(f_3) à l'AIRLNDPM(f_1).
Le fait que ces courbes se chevauchent indique que l'AIRLNDM(f_3) est capable de donner d'aussi bons résultats que l'AIRLNDPM(f_1).
Les évolutions obtenues pour les autres instances sont similaires à celles de ft10. Nous observons cependant une dégradation des performances de l'AIRLNDM(f_3) en fonction de la taille de l'instance contrairement à l'AIRLNDPM(f_1). Ceci tend à réduire le nombre de marches pour lesquelles l'AIRLNDM(f_3) égale ou surpasse l'AIRLNDPM(f_1) en terme de makespan de la solution finale. Cependant, cet effet est compensé par le fait que la durée d'exécution de l'AIRLNDPM croît beaucoup plus vite que celle de l'AIRLNDM en fonction de la taille de l'instance. Par conséquent, il est possible de lancer plusieurs exécutions de l'AIRLNDM pour une durée totale comparable à la durée d'une seule exécution de l'AIRLNDPM. En résumé, à durée d'exécution égale (ou pour un même nombre d'évaluations), plusieurs exécutions de l'AIRLNDM(f_3) sont susceptibles de donner des résultats similaires à une exécution de l'AIRLNDPM(f_1).

7.3.3.5 Discussion

Les critères retenus (H_2 et C_{op}) demandent peu de calculs supplémentaires puisque les opérations critiques sont implicitement déterminées par l'opérateur lors de la phase de réordonnancement, de même que les valeurs de $E^m(x)$ nécessaires au calcul de H_2. Le seul coût (au sens informatique du terme) induit par l'ajout des critères est lié au calcul de la somme des $(E^m(x))^2$ sur l'ensemble des machines et au calcul de la fonction coût en elle même (qui se traduit par deux multiplications et deux additions pour f_3). Par conséquent, le coût engendré par les calculs nécessaires aux fonctions multicritères retenues est négligeable. Quel que soit le gain (en terme de makespan) obtenu, ce dernier n'induit aucun surcoût en terme de temps processeur pour un même nombre d'appels à la fonction d'évaluation. Par contre, le fait d'introduire des critères secondaires allonge généralement les marches des AIRL, ce qui se traduit par un nombre supérieur d'appels à la fonction d'évaluation. C'est en ce sens que l'on peut mesurer le coût (en terme de temps processeur ou de nombre d'évaluations ou d'itérations) induit par l'ajout des critères secondaires.

Le gain relatif maximum entre f_1 et f_3 est obtenu pour l'AIRLD : il est de l'ordre de 2% sur les instances considérées si l'on compare les performances moyennes entre AIRLD(f_3) et AIRLD(f_1).

Nous avons surtout insisté sur les critères (H_2 et C_{op}) pour illustrer notre démarche. Cependant, il existe bien d'autres critères qui mériteraient une étude complémentaire. C'est le cas par exemple du critère C_2 défini dans le cadre de nos travaux et utilisé dans la fonction f_6 : comme le montre la figure 7.10 (page 111), ce critère permet à f_6 de surpasser les autres fonctions sur certaines instances (toutefois le gain relatif entre f_1 et f_6 pour l'AIRLD reste de l'ordre de 2% sur les instances considérées).

En ce qui concerne l'AIRLND, l'intégration d'une mémoire élémentaire nous a permis d'améliorer les résultats obtenus, tout en réduisant le nombre d'appels à la fonction coût.

Pour les différents AIRL, l'utilisation du critère H_2 est justifiée. Par contre, les résultats sont beaucoup plus nuancés en ce qui concerne l'utilisation du nombre d'opérations critiques C_{op}.

L'AIRLNDPM montre que nos fonctions multicritères sont incompatibles avec un mécanisme de déplacement sur les plateaux. Ces fonctions apportent un gain aux AIRL tant qu'ils ne se déplacent pas sur les plateaux. Ces remarques permettent de restreindre le domaine d'utilisation de telles fonctions aux AIRL intégrés dans des méthodes hybrides. Pour ces méthodes, le but de l'AIRL est d'offrir un bon compromis entre vitesse d'exécution et qualité de solution finale. Or, d'une part l'AIRL utilisant les fonctions multicritères offre de meilleures performances qu'un AIRL basé exclusivement sur l'évaluation du makespan. D'autre part, l'AIRLNDPM présente un inconvénient qui n'est pas clairement ressorti de l'étude car il ne prend d'importance que lorsque la durée d'exécution entre en jeu. En effet, l'AIRLNDPM peut prendre un temps non négligeable à se déplacer sur un plateau sans parvenir à améliorer la solution courante, ce qui n'est pas acceptable lorsque l'AIRLNDPM est utilisé en guise d'opérateur dans une méthode hybride : pendant que l'AIRLNDPM cherche une meilleure solution sur le plateau en bloquant éventuellement la méthode hybride, cette dernière aurait pu se montrer plus efficace en « contournant » le plateau.

Par conséquent, pour une méthode non hybride, ou pour une hybridation de type parallèle asynchrone ou encore une hybridation séquentielle, un AIRLNDPM est tout à fait utilisable. Surtout lorsque l'on considère les performances obtenues sur les instances de la *OR-Library*. Dans ce cas l'utilisation des fonctions multicritères telles que nous les avons définies n'est pas justifiée.

En revanche, pour une méthode hybride de type parallèle synchrone, un AIRL dont le mécanisme de déplacement sur les plateaux est remplacé par l'utilisation de fonctions multicritères offre un bon compromis entre durée d'exécution et qualité de solution finale.

Lors de la définition des fonctions multicritères, nous nous étions posé la question de l'utilité

des déplacements sur les plateaux : est-il préférable de stopper un AIRL piégé par un plateau pour lancer un AIRL depuis une solution initiale différente, ou au contraire de tenter de se déplacer sur le(s) plateau(x) dans l'espoir de découvrir une meilleure solution ?

Une réponse à cette question peut être obtenue en considérant le nombre de plateaux dans l'espace de recherche, leur taille, ainsi que leur dispersion dans cet espace (sont-ils en périphérie d'un massif central ou dispersés dans l'espace de recherche ?). Dans le cas où il y a peu de plateaux dispersés dans l'espace, la probabilité de « rencontrer un plateau » est faible. Par conséquent, il est inutile de prévoir un mécanisme particulier de déplacement sur les plateaux, il suffit de relancer l'AIRL depuis une autre solution initiale. Par contre, s'il y a beaucoup de plateaux dispersés, la probabilité de rencontrer un plateau lors du cheminement de l'AIRL est très élevée, et il est préférable de prévoir un mécanisme de déplacement sur les plateaux ou un mécanisme permettant de réduire la taille des plateaux. La taille des plateaux quant-à elle va déterminer la difficulté (en terme de nombre de mouvements) avec laquelle l'AIRL pourra s'échapper d'un plateau.

Or l'étude du paysage du JSP (section 7.1.1.1, page 81) révèle la présence de nombreux plateaux et optima locaux dispersés dans l'espace de recherche, ce qui conforte l'idée d'un mécanisme de déplacement sur les plateaux. Cependant, l'AIRLNDPM montre que les marches sur les plateaux peuvent être coûteuses (en terme de nombre d'évaluations ou de durée d'exécution). Dans le cas de plateaux de « grande taille », il semble plus intéressant de relancer un AIRLNDM plutôt que de laisser évoluer un AIRLNDPM. Donc pour répondre à la question posée ci-dessus, étant donné la présence de nombreux plateaux répartis dans l'espace du JSP, l'utilisation d'un AIRLNDPM est justifiée si la taille des plateaux est suffisamment faible pour que l'AIRLNDPM puisse s'en échapper en un temps raisonnable. Malheureusement, la taille des plateaux semble difficile à appréhender : certes elle est une conséquence de l'imprécision du makespan, de la taille de l'instance et de l'opérateur utilisé ; mais le calcul de la taille (et du nombre) des plateaux est loin d'être évident. Même si certains critères tels que les opérations critiques sont liés à la présence de plateaux et à la difficulté d'une instance, l'ensemble des critères qui déterminent la difficulté « *a priori* » d'une instance de jobshop est loin d'être facile à appréhender comme le montrent les résultats de la section 7.2.

7.3.4 Application à d'autres problèmes

Pour le problème du *jobshop*, il existe un grand nombre de critères secondaires pour mesurer la qualité d'un ordonnancement. Cependant, cette méthode est applicable à bien d'autres problèmes pourvu de consacrer suffisamment de temps à la recherche de ces critères secondaires.

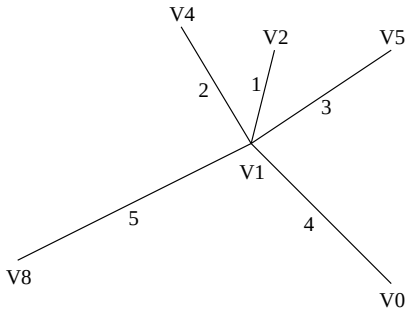


Figure 7.21: Rang d'un arc pour le TSP : le rang d'un arc issu d'une ville donnée (V1 sur la figure) est la position de cet arc lorsque l'on trie les arcs par longueur croissante. Dans l'exemple ci-dessus, la valeur du rang est indiquée sur chaque arc.

Nous pouvons par exemple appliquer cette méthode au voyageur de commerce (TSP) en utilisant comme critère secondaire le « rang » des arcs : le rang d'un arc issu d'un sommet donné est la position

de cet arc lorsque l'on trie les arcs par longueur croissante (voir figure 7.21). Ce critère est basé sur le fait qu'un chemin court est composé d'un grand nombre d'arcs courts. [SW87] évoquent la mise en œuvre d'un principe similaire en utilisant comme critère secondaire une fonction basée sur le nombre de villes connectées à leur plus proche voisine.

[JS89, SD90, DH94, GV97] ont appliqué un principe analogue pour résoudre un problème de type SAT, en définissant les équations logiques sous forme normale conjonctive et en intégrant des informations complémentaires dans la fonction coût. Leur fonction coût n'est pas directement basée sur une fonction multicritère composée d'un critère principal et de critères secondaires, mais sur l'intégration directe de ces critères via une modification de la fonction coût.

Il s'avère possible d'intégrer des critères complémentaires dans les fonctions coût de différents problèmes, y-compris pour des problèmes tels que SAT, où il n'existe initialement aucune information complémentaire dans la fonction coût. De même, il semble possible de trouver et d'intégrer de telles informations pour n'importe quel problème, ne serait-ce que des informations liées à des heuristiques typiquement utilisées pour accélérer la recherche de solutions acceptables.

7.3.5 Discussion

Les résultats montrent que les performances des AIRL ne sont pas directement liées à la taille de l'instance. En fait, l'augmentation de la taille de l'instance agrandit le voisinage de chaque point de l'espace. De ce fait, le nombre d'alternatives dont dispose l'AIRL pour choisir le prochain point à visiter augmente, ce qui accroît la probabilité de quitter un plateau. Pour un opérateur donné, le problème est de déterminer si la taille des plateaux n'est pas elle aussi en rapport avec la taille de l'instance.

Inst	f_1		f_2		f_3	
	<pas>	$\%_{\text{éval}}$	<pas>	$\%_{\text{éval}}$	<pas>	$\%_{\text{éval}}$
ft10	1044.83	21.47	550.91	21.47	450.64	18.93
ft20	887.96	14.43	216.45	11.00	198.78	10.54
ta01	3779.68	24.23	949.29	21.44	1028.15	22.26
abz7	6994.92	21.93	2099.16	19.58	1987.60	19.63
ta31	4983.38	16.52	698.89	13.23	672.09	13.01

Tableau 7.13: Étude du déplacement sur les plateaux. La colonne <pas> donne le nombre de pas effectués en moyenne sur les plateaux. Cette mesure reflète, de manière très approximative, l'évolution de la taille des plateaux en fonction de la taille des instances étudiées. La colonne $\%_{\text{éval}}$ présente le pourcentage d'évaluations utilisées pour les déplacements sur les plateaux (par rapport au nombre total d'évaluations). Une valeur constante ou croissante pour ce paramètre indiquerait que la taille des plateaux est corrélée à la taille des instances. Aucune corrélation ne peut être clairement établie entre la taille des instances et les mesures présentées. Ce résultat suggère l'absence d'une forte corrélation entre la taille des instances et la taille des plateaux, sans pour autant affirmer qu'il n'existe absolument aucune relation entre ces dernières.
Les résultats des fonctions multicritères f_2 et f_3 indiquent une réduction du nombre de pas sur les plateaux, ce qui confirme indirectement la réduction de la taille des plateaux induite par ces fonctions.

Les résultats obtenus nous apportent quelques éléments de réponse qui semblent indiquer qu'il n'y a pas de lien direct entre la taille des plateaux, le nombre de plateaux, et la taille de l'instance (voir tableau 7.13). Pour illustrer notre propos, voici deux exemples extrêmes où la taille de l'instance n'intervient pas directement sur le nombre de plateaux :

- une instance $J \times M$ telle que $\forall j, m \ \ 1 \leq j \leq J, 1 \leq m \leq M \ \ d(O_j^m) = 2^{(m-1)J+(j-1)}$ ou $d(O_j^m) = 2^{(j-1)M+(m-1)}$ ne possède pas de plateau et contient un seul chemin critique quel que soit

l'ordonnancement considéré ;

– au contraire, une instance telle que $J \gg M$, pour laquelle l'écart-type de $d(O_j^m)$ est faible comporte un grand nombre de solutions de même makespan (cette instance tend vers le cas du JSP préemptif lorsque l'écart-type tend vers zéro).

Dans cette section, nous nous sommes concentrés sur l'utilisation de fonctions coût multicritères au sein des algorithmes itératifs de recherche locale. Nous nous sommes intéressés plus particulièrement au problème qui survient lorsque l'AIRL atteint un plateau dans l'espace de recherche : en l'absence de critères secondaires l'AIRL est incapable d'obtenir des informations susceptibles de le guider efficacement vers le prochain point à visiter. Afin de résoudre ce problème, nous avons proposé une méthode basée sur des critères secondaires capables d'apporter des informations complémentaires sur la qualité – en terme de critère principal – de la solution. Nous avons intégré cette technique dans plusieurs algorithmes itératifs de recherche locale et montré son utilité dans le cadre de la résolution du *jobshop* simple.

De manière à montrer que cette approche peut être utilisée dans différents types d'AIRL, nous nous sommes basés sur le même ensemble de fonctions coût dans un algorithme évolutif (AE) et nous l'avons appliqué au même ensemble d'instances. Cette technique présente un avantage supplémentaire lorsqu'elle est appliquée aux AE : l'ajout d'informations complémentaires dans la fonction coût induit une augmentation de la diversité de la population d'individus, ce qui se traduit par un accroissement des performances (les résultats sont présentés dans le tableau 8.2, page 143).

La comparaison des résultats de l'AIRLNDM à ceux de l'AIRLNDPM montrent que les fonctions multicritères et le cheminement sur les plateaux ne sont pas compatibles entre eux lorsqu'ils sont implantés au sein d'une même heuristique. Ceci peut avoir deux causes :

1. soit le cheminement sur les plateaux des fonctions multicritères n'est pas souhaitable et engendre des solutions de mauvaise qualité du point de vue du critère principal ;
2. soit le cheminement sur les plateaux a été interrompu trop tôt par le critère d'arrêt.

Pour la première hypothèse, le cheminement sur les plateaux du critère principal donne de bons résultats lorsqu'il n'est pas guidé par les critères secondaires. Lorsque le déplacement est autorisé sur les plateaux de la fonction coût (multicritère ou non), l'ajout de critères secondaires détériore les performances en moyenne. Cela pourrait s'expliquer par le fait que l'amplitude bruit engendré par le déplacement sur les plateaux est supérieure au gain apporté par les critères secondaires utilisés. En ce cas, des critères secondaires plus performants devraient améliorer les résultats obtenus pour les fonctions multicritères. Certains critères secondaires semblent remplacer les plateaux par une fonction qui engendre une rugosité supplémentaire, ce qui ajoute un certain nombre de « faux » optima locaux dans le paysage.

La seconde hypothèse peut être immédiatement rejetée en considérant les courbes présentées sur la figure 7.19. En effet, hormis pour ft20, ces courbes montrent que l'AIRLNDPM(f_1) donnent de meilleurs résultats que l'AIRLNDPM(f_2) et l'AIRLNDPM(f_3) : à aucun moment l'AIRLNDPM(f_2) ou l'AIRLNDPM(f_3) ne s'approche des performances de l'AIRLNDPM(f_1). La tendance des courbes lors de l'arrêt de l'AIRLNDPM ne laisse pas entrevoir d'amélioration des performances de l'AIRLNDPM(f_2) ou l'AIRLNDPM(f_3) en extrapolant les courbes. Autrement dit, un nombre supérieur d'itérations sur les plateaux n'améliorerait pas les performances de l'AIRLNDPM(f_2) ou de l'AIRLNDPM(f_3). Ce résultat est conforté par le fait que la taille des plateaux est réduite lors de l'utilisation des fonctions multicritères (voir tableaux 7.13). Par conséquent le critère d'arrêt, basé sur un nombre fixé d'évaluations avant amélioration favorise les fonctions multicritères contrairement à l'hypothèse avancée.

7.4 Voisinage non constant

L'utilisation d'un voisinage non constant est une méthode qui permet d'explorer l'espace de recherche de manière efficace : soient deux couples ⟨codage1, opérateur1⟩ et ⟨codage2, opérateur2⟩, alors un plateau ou un optimum local dans l'espace de recherche défini par ⟨codage1, opérateur1⟩, n'est pas nécessairement un optimum local ou un plateau dans l'espace de recherche relatif à ⟨codage2, opérateur2⟩. Par conséquent, le passage d'un couple à l'autre au cours de la recherche peut s'avérer particulièrement utile pour éviter les pièges que sont les optima locaux et les plateaux.

Il existe différentes méthodes pour mettre en œuvre cette technique, parmi lesquelles :

- l'utilisation simultanée ou consécutive de différents codages [FRP97b] ;
- la variation de la taille d'un voisinage restreint au cours du temps [HW95, NS96b] ;
- l'utilisation simultanée ou consécutive de plusieurs méthode de recherche.

Le premier point nécessite, en plus de la définition de plusieurs codages et des opérateurs associés, la mise en place d'un ensemble de mécanismes de transcodage permettant de transcrire les solutions selon les différents codages. Il peut être intéressant d'utiliser simultanément plusieurs codages si, par exemple, un opérateur s'exprime plus simplement lorsqu'il agit sur un premier codage, alors qu'un autre opérateur utilisé simultanément s'exprime plus facilement lorsqu'il agit sur un autre codage.

Le second point est largement utilisé dans la recherche tabou en raison de l'extrême simplicité de l'implantation de cette méthode : il suffit en effet de faire varier la taille de la liste tabou au cours du temps pour obtenir une taille de voisinage variable.

Le dernier point correspond à l'utilisation de méthodes hybrides. L'idée sous-jacente consiste à changer de méthode lorsqu'une méthode donnée ne progresse plus. Bien souvent chaque méthode de recherche utilise son propre codage ; de ce fait, la méthode hybride résultante renferme plusieurs codages et vraisemblablement différents opérateurs de déplacement. Ce dernier point a pour conséquence qu'un plateau « vu » par une première méthode de recherche ne correspond pas nécessairement à un plateau pour une autre méthode de recherche utilisée conjointement, ou n'est pas traité[25] de la même manière par ces méthodes.

Nous avons expérimenté ces techniques lors de l'application de méta-heuristiques au jobshop.

[25] du point de vue du déplacement dans la zone contenant ce plateau.

Chapitre 8

Application des méta-heuristiques au *jobshop*

Dans le cadre de notre étude, nous nous intéressons à l'application des méta-heuristiques à la résolution de problèmes d'optimisation. Nous étudions les méta-heuristiques dans leur forme la plus simple afin de comprendre leur fonctionnement. Nous ne cherchons pas à paramétrer ou à modifier les méta-heuristiques de manière à obtenir les meilleures performances possibles, car en général ces modifications induisent un comportement difficilement analysable des méthodes de recherche. De ce fait, nous proposons plusieurs techniques d'amélioration des performances en utilisant divers outils, tels que les paysages adaptatifs, pour expliquer leur fonctionnement. Grâce à ces outils, nous analysons les composants élémentaires de plusieurs méta-heuristiques, en nous concentrant plus particulièrement sur les algorithmes évolutifs et la recherche tabou. Notre choix s'est porté sur ces deux méthodes de recherche en raison, d'une part, de leur caractéristiques intrinsèques :

- même dans sa version la plus simple, la recherche tabou est très performante lorsqu'il s'agit d'exploiter une zone particulière de l'espace de recherche ;
- les algorithmes évolutifs sont de « redoutables explorateurs » d'espaces de recherche.

D'autre part, ces méta-heuristiques ont d'ores-et-déjà été utilisées avec succès dans de nombreux domaines. Ces deux méthodes permettent de concevoir des algorithmes hybrides figurant parmi les méthodes de résolution les plus efficaces à ce jour. Dans ces associations, l'algorithme évolutif est généralement chargé d'explorer, tandis que la recherche tabou exploite l'espace de recherche. Bien que ces méthodes soient fréquemment utilisées, les études sur le comportement des composants de base de ces méthodes sont rares, et les choix effectués sont bien souvent empiriques. Prenons l'exemple de la fonction coût : en général, cette fonction est directement la valeur que l'on cherche à optimiser même s'il existe, pour le problème considéré, des informations complémentaires capables de guider la méta-heuristique dans sa recherche d'une solution.

Partant de cette constatation, nos travaux portent sur l'analyse des composants de base de ces méta-heuristiques. Pour mener à bien cette analyse, nous utilisons un ensemble d'outils et de critères, dont certains sont directement issus de nos travaux. La difficulté de cette analyse réside dans les interdépendances entre les différents composants qui rendent inutile toute approche réductionniste du problème. Nous utilisons un problème de type *jobshop* pour illustrer nos travaux sur un exemple concret, cependant les outils mis en œuvre sont utilisables sur une large gamme de problèmes.

La démarche suivie pour mettre en œuvre nos analyses se décompose en plusieurs points :

Tout d'abord, nous avons étudié le codage des solutions : [BUMK91, Bru93, ERR94, GV97] montrent qu'il est préférable d'intégrer un maximum d'informations sur le problème traité dans le codage et les opérateurs. Pourtant, le choix du codage et des opérateurs est bien souvent dicté par la simplicité du code nécessaire à leur implantation. Aussi nous comparons différents codages plus ou moins complexes pour déterminer notre choix. Nous comparons également différents opérateurs associés à ces codages.

Une fois les couples ⟨codage, opérateurs⟩ définis, nous étudions le paysage du problème traité. Les outils relatifs à l'étude du paysage fournissent des informations statistiques sur la répartition des optima locaux et des plateaux dans l'espace de recherche. Le problème du *jobshop* nous permet de définir différents voisinages restreints, tout en garantissant l'existence de chemins aboutissant à une solution optimale. De ce fait, nous étudions le comportement des méta-heuristiques lors de l'utilisation de ces différents voisinages.

Ainsi que cela a été évoqué précédemment, la fonction objectif est bien souvent restreinte au calcul du coût lié au problème à résoudre, au mépris d'informations complémentaires qui peuvent être utiles pour guider les méta-heuristiques dans l'espace de recherche. Pour appliquer cette technique au *jobshop*, nous définissons plusieurs fonctions multicritères, et nous analysons leur impact sur le paysage adaptatif. Les critères secondaires introduits dans nos différentes fonctions coût permettent de réduire les « imprécisions » de la fonction coût dans une certaine mesure. Ceci revient à réduire le nombre de solutions voisines possédant un même coût. Cependant, ces fonctions ne suppriment pas tous les plateaux[1] dans l'espace de recherche. Aussi, nous définissons deux types de méthodes heuristiques : les heuristiques « capables de se déplacer sur les plateaux » et les heuristiques « non autorisées à se déplacer sur les plateaux ».

Une fois ces différents points étudiés, nous les appliquons à la recherche tabou : nous étudions notamment l'apport induit par l'utilisation de voisinages restreints non constants au cours de la recherche [HM97].

Intégrée dans un algorithme évolutif, une fonction multicritère induit une augmentation de la diversité de la population. Nous analysons l'impact de cette augmentation de diversité sur les performances de l'algorithme évolutif.

L'opérateur de recombinaison est l'une des différences principales des algorithmes évolutifs par rapport aux autres méthodes de recherche. Aussi nous étudions l'efficacité de cet opérateur. La présence de plusieurs opérateurs dans les algorithmes évolutifs nécessite la définition d'un schéma d'application de ces opérateurs. En règle générale, un schéma séquentiel est utilisé, c'est-à-dire que l'opérateur de mutation est appliqué aux individus engendrés par l'opérateur de recombinaison. Nous définissons un schéma d'application parallèle qui offre de meilleures performances lors de l'utilisation d'une certaine classe d'opérateurs que nous caractérisons.

Pour terminer, nous hybridons un algorithme évolutif et une recherche tabou afin de tirer profit de leur caractéristiques intrinsèques (exposées précédemment). Les résultats montrent que les instances du *jobshop* sont très hétérogènes du point de vue de leur « difficulté » ; cette dernière n'est pas directement liée à la taille de l'instance, aussi nous définissons un ensemble de critères qui nous renseignent *a priori* sur cette difficulté.

À l'issue de l'étude de ces différents composants, nous proposons diverses techniques pour améliorer les performances des méta-heuristiques étudiées. Ensuite, nous présentons divers schémas d'hybridation. Enfin, nous terminons ce chapitre par une comparaison des résultats obtenus. Avant de commencer l'étude, la section suivante précise la démarche retenue pour la comparaison de nos résultats à d'autres auteurs.

[1] Ensemble de solutions voisines possédant un même coût (voir définition 60, page 99).

8.1 Comparaisons des performances

Cette section est consacrée à la comparaison des performances des méthodes de résolution appliquées au *jobshop*.

La comparaison entre méthodes de recherche se heurte non seulement à des problèmes liés aux fonctionnements intrinsèquement différents, mais aussi à des problèmes davantage liés aux méthodologies utilisées pour réaliser les tests, ou encore à la disponibilité des informations relatives aux paramétrages utilisés. Voici quelques uns des problèmes rencontrés lors de la collecte d'informations :

1. plan d'expériences imprécis concernant le nombre d'expériences lancées, le type de solutions initiales (aléatoires ou non), l'utilisation des mêmes solutions initiales (ou de solutions différentes) pour comparer les méthodes ;
2. utilisation d'instances créées par l'auteur et non rendues disponibles ;
3. résultat d'une méthode donné sous forme d'une valeur sans préciser s'il s'agit d'une valeur minimale, maximale ou moyenne ;
4. résultat minimum présenté sans préciser la fréquence d'obtention de cette valeur ;
5. moyenne des résultats donnée sans la valeur de l'écart-type (ni aucune information sur la répartition des valeurs de coût des solutions obtenues) ;
6. résultats comparatifs entre deux méthodes sans précision sur le nombre d'évaluations effectuées pour chacune d'entre-elles ;
7. résultats annoncés comme issus d'une méthode de recherche alors qu'il s'agit en réalité d'une méthode hybride ;
8. résultats imprécis en ce qui concerne le paramétrage, le codage ou les opérateurs utilisés.

Les critères de comparaison et surtout les conditions dans lesquelles sont effectuées ces comparaisons varient fortement d'un auteur à l'autre [Tai93b, JW94, KKN93, MH93a, VAL92, TCM95, HW96, Par95, IHI96, NS96a, YN96, FRC94, TN92, DG96].

Notre objectif principal n'est pas la comparaison des résultats de différentes méthodes de résolution, ni la recherche des performances optimales. Cependant de manière à ce que nos résultats puissent être facilement comparés à d'autres auteurs, nous utilisons les instances du *jobshop* issues de la *OR-Library* [Bea90]. L'utilisation de la *OR-Library* offre un autre avantage : une liste des meilleures performances mondiales est disponible sur le site WWW ; elle est mise à jour périodiquement suite à l'amélioration d'un résultat ou à l'arrivée de nouvelles instances.

Nous avions entrepris une comparaison de résultats entre différents auteurs (voir annexe B, page 215), mais dans le cadre de notre étude l'utilisation de la *OR-Library* s'est avérée suffisante.

Nous ne donnons pas d'information concernant la durée d'exécution (excepté à l'annexe A.3.1, page 182) car là encore notre objectif n'est pas la recherche d'une implantation optimale de nos algorithmes sur les calculateurs. De plus, il peut s'avérer dangereux de comparer directement la durée d'exécution des méthodes de recherche (une méthode peut progresser vite, mais se précipiter vers un optimum local). [BDF97] recensent un certain nombre de pièges pouvant conduire à des résultats erronés lors de la comparaison de méthodes de résolution.

8.2 Couples <codage, opérateur>

L'application d'une méta-heuristique à un problème donné passe par la définition d'un couple ⟨codage, opérateur⟩. Dans cette section, nous passons en revue quelques codages et opérateurs applicables au JSP.

8.2.1 Codages

De nombreux codages ont été proposés pour résoudre les problèmes de type *jobshop* si bien qu'il est difficile d'en faire une liste exhaustive. Par conséquent, nous présentons ces codages en les regroupant par « familles » en fonction de leurs caractéristiques.

La première distinction concerne le type de *jobshop* traité : certains codages sont exclusivement dédiés au *jobshop* simple, d'autres sont exclusivement dédiés à des formes généralisées de *jobshop*. Enfin, il existe des codages applicables au *jobshop* simple comme aux *jobshops* généralisés. Nous synthétisons les différentes familles de codages dans le tableau 8.1.

Type de codage	Références
Codage à base de vecteurs d'entiers (un entier représente un numéro d'heuristique appliquée à un produit ...)	[Por96, FRC93, HR98]
Codage à base de tableau de marqueurs (un entier représente un numéro de produit (*marqueur*) ...)	[BMK96, Por96]
Codage à base de permutations d'entiers (un entier représente un numéro d'opération ou un numéro de produit ...)	[Bru93, BUMK91, DPT95b, Evo97, BMK96, Por96, UBKM93]
Codage sous forme de chaînes de bits (un bit représente l'arbitrage d'une paire de disjonction ou la priorité entre deux opérations ...)	[NY91, Evo97, BMK96, TN92]
Codage sous forme de matrice binaire (un élément représente la précédence entre deux opérations ...)	[Por96]
Matrice de séquences, matrices de permutations (non binaires)	[TCM95, Por96, DPT96b, KOY95, DPT$^+$98]
Codages composites linéaires, constitués d'un tableau d'éléments structurés (tableau de paires <produit,plan> ...)	[Bru93, BUMK91, DPT95b, TCM95, UBKM93, Soa94, DPT95c]
Codages composites non linéaires, constitué de plusieurs tableaux (un élément représente un numéro d'heuristique appliquée à un produit, ou un numéro d'opération ...)	[PG95, Por96, Dav85, ABN93, FB91, Ghe95]
Codages composites matriciels et codages directs, constitués d'une matrice d'éléments structurés (matrice de tuples <produit,plan,machine,début,fin> ...)	[Bru93, DPT96b, DPT95b, Evo97, Por96, UBKM93, DPT95c, DPT96b, DPT96a, DPT$^+$98, LGWFP97]

Tableau 8.1: Quelques familles de codages dédiés au *jobshop*.

8.2.1.1 Codages retenus

Nous étudions deux codages particuliers :

- un codage indirect basé sur des tableaux de marqueurs ;
- un codage direct.

Nous avons retenu un codage indirect sous forme de tableau de marqueurs car il s'agit d'un codage simple qui ne nécessite pas de mécanisme de réparation ni de décodeur complexe pour obtenir un ordonnancement réalisable. Les opérateurs agissant sur ce codage sont également assez simples à réaliser. Ce codage ne comporte pas que des avantages, son principal inconvénient est

d'être redondant. Le principe du codage est le suivant : un marqueur de valeur k est associé au produit k. Le tableau de marqueurs est lu de gauche à droite et à chaque fois qu'un marqueur de valeur k est rencontré, la prochaine opération non effectuée du produit k est réalisée (en respectant les contraintes de précédence et de ressource). Un exemple simple permet d'illustrer la redondance du codage (voir figure 8.1).

N°opération	0	1	2	3	4	5
N°machine	1	0	0	1	0	1
N°produit	0	0	1	1	2	2
N°marqueur	0	0	1	1	2	2
Durée	-	-	-	-	-	-

Caractéristiques des opérations

T_1	0	1	0	2	1	2
T_2	1	0	0	2	1	2

Tableaux de marqueurs

Machine	Opérations		
0	2	1	4
1	0	3	5

Matrice de séquences

Figure 8.1: Exemple de codage sous forme de tableaux de marqueurs : T_1 et T_2 sont deux tableaux de marqueurs d'une instance de JSP3×2 ; les produits 0, 1 et 2 sont associés à deux marqueurs (1 par machine) de valeurs respectives 0, 1 et 2. Si les deux premières opérations 0 et 2 sont réalisées sur des machines différentes, alors les deux tableaux de marqueurs T_1 et T_2 correspondent à la même matrice de séquences (et par conséquent au même ordonnancement à l'issue du décodage) alors qu'ils sont différents.

Le second codage utilisé – illustré par la figure 8.2 – est un codage direct de type « composite matriciel ». Son principal avantage est de fournir instantanément toutes les informations sans nécessiter de décodage, ce qui permet de définir des opérateurs complexes. Ses inconvénients sont liés à cette richesse d'information : il est plus coûteux en mémoire et nécessite la définition d'opérateurs plus complexes que les opérateurs associés au premier codage.

machine 1	Op8 Job1 <1,10>	...	Op1 Job2 <51,55>	...
⋮	⋮	⋮	⋮	⋮
machine M	Op5 Job1 <11,31>	...	Op4 Job2 <34,36>	...

Figure 8.2: Exemple de codage direct : pour chaque machine un tableau de périodes est défini, chaque période contient toutes les informations relatives à l'exécution d'une opération (i.e dates de début et de fin notées <début,fin>, numéro de produit, numéro d'opération).

L'utilisation de ces deux codages permet de comparer les possibilités offertes par un codage indirect aux possibilités offertes par un codage direct. Le codage indirect sera utilisé lorsque des opérateurs simples et rapides suffisent à l'étude d'un critère particulier (pour certains calculs basés sur des marches d'AIRL par exemple). Le codage direct sera utilisé lorsque des opérateurs plus complexes, et donc supposés plus performants, doivent être utilisés.

8.2.2 Opérateurs

Les informations héritées via les opérateurs jouent un rôle important au niveau des performances de la méthode de recherche. Ainsi que cela a été évoqué en section 3.2.1 (page 26), les informations à transmettre pour obtenir des opérateurs performants diffèrent selon le problème traité. Par exemple, ce qui doit être hérité par crossover pour résoudre efficacement le TSP ne s'applique pas au *jobshop* : lorsqu'un codage basé sur une permutation d'entiers est utilisé pour résoudre le *jobshop* simple,

l'ordre absolu est plus important que l'ordre relatif [SDMW96, NS96a]. Par exemple, l'opérateur de recombinaison « edge recombination » (ER) fonctionne bien pour le TSP mais pas pour le *jobshop* [WSF89].

8.2.2.1 Opérateurs retenus

Nous avons défini un ensemble d'opérateurs pour chaque codage précédemment défini.

En codage indirect, nous nous sommes limités à la définition d'un opérateur de mutation. Cet opérateur consiste à permuter deux marqueurs de valeur k et k', tels que $k \neq k'$.

En codage direct, nous avons défini plusieurs opérateurs de mutation. Certains sont basés sur la notion d'opération critique. Notre algorithme évolutif utilise un crossover GA/GT.

Ces opérateurs ont été mis au point grâce à l'outil *Visusch* développé dans le cadre de nos travaux (voir section D.2, page 224).

(C_1) effectuer les étapes ① et ② de l'algorithme GT[2] afin d'obtenir C, EC et G :

① déterminer l'opération O_{j*}^{m*} telle que $EC(O_{j*}^{m*}) = \min\{EC(O_j^m) \mid O_j^m \in C\}$.

② Constituer $G = \{O_j^{m*} \in C$ telles que les exécutions de O_j^{m*} et O_{j*}^{m*} se recouvrent$\}$.

(C_2) choisir l'une des opérations de l'ensemble G (en vue de l'ordonnancer) de la manière suivante :

ⓐ générer un nombre aléatoire $\varepsilon \in [0,1]$ et le comparer à R_μ.
si ($\varepsilon < R_\mu$) **alors** choisir une opération O_{j*}^{m*} depuis G (la mutation survient).

ⓑ **sinon** individu parent ← (sélectionner « père » ou « mère » avec une égale probabilité de $\frac{1}{2}$).
Chercher O_{j*}^{m*} ordonnancée le plus tôt dans « parent » parmi toutes les opérations $\in G$:
$O_{j*}^{m*} = \min\{\text{parent}_j^m \mid O_j^m \in G\}$.

ⓒ ordonnancer O_{j*}^{m*} au plus tôt (*i.e.* selon $EC(O_{j*}^{m*})$), puis initialiser $\text{fils}_{j*}^{m*} = EC(O_{j*}^{m*})$.

(C_3) mettre à jour C et EC.

Itérer ($C_1 C_2 C_3$) jusqu'à ce que toutes les opérations soient ordonnancées pour obtenir le nouvel individu fils.

Figure 8.3: Crossover GA/GT. Chaque individu correspond à un ordonnancement actif. L'algorithme présenté génère un « fils » à partir de 2 individus appelés « père » et « mère ». Pour obtenir deux nouveaux individus « fils1 » et « fils2 », il suffit de répéter deux fois ces étapes sur la même paire d'individus « père » et « mère ». Une autre méthode consiste à générer deux individus fils en parallèle lors de l'étape C_2.
$R_\mu \in [0,1]$ est une constante prédéfinie appelée « taux de mutation interne ».

Opérateur de recombinaison GA/GT Tel qu'il est défini par [NY92], cet opérateur est assez complexe. Basé sur l'algorithme de Giffler et Thompson (GT)[2], l'opérateur de recombinaison « GA/GT » est en réalité constitué d'un opérateur de recombinaison couplé à un opérateur de mutation interne (voir figure 8.3). Selon le taux d'application de l'opérateur de mutation interne, cet opérateur peut provoquer une chute rapide de la diversité de la population. En fait, il nécessite une grande taille de population pour fonctionner correctement. Ainsi [NDY94] ont évalué la taille optimale de la population lors de l'utilisation de l'opérateur de recombinaison GA/GT.

8.2.2.2 Ergodicité et accessibilité de la solution optimale

Dans cette section, nous étudions quelques propriétés des opérateurs utilisés dans le cadre de nos travaux.

[2]Voir figure 6.1, page 74

Mutation en codage indirect L'opérateur de mutation utilisé en codage indirect permute deux marqueurs de valeurs différentes. Il est ergodique[3] car il permet d'atteindre tout point de l'espace de recherche $\mathcal{E}$ constitué de l'ensemble des tableaux de marqueurs. Cependant, ainsi que cela a été évoqué en section 3.1.1, le décodeur ne doit pas interdire la découverte d'une solution optimale. De ce fait, nous utilisons un décodeur qui ordonnance toujours les opérations selon l'ordre imposé par le tableau de marqueurs et qui transforme ensuite l'ordonnancement obtenu en ordonnancement semi-actif sans modifier l'ordre des opérations (autrement dit sans effectuer de décalages à gauche[4]). À l'issue du décodage, seuls des ordonnancements semi-actifs sont engendrés. Par conséquent le couple $\langle$mutation, décodeur$\rangle$ ne permet pas d'atteindre tout point de l'espace. Le décodeur ne modifie pas l'ordre imposé par le tableau de marqueurs auquel il est appliqué et l'opérateur de mutation est ergodique. Donc tout ordonnancement semi-actif peut être engendré par le couple $\langle$mutation, décodeur$\rangle$. Or, les solutions optimales sont des ordonnancements actifs et les ordonnancements actifs sont semi-actifs. Et comme tout ordonnancement semi-actif peut être engendré, tout ordonnancement actif est accessible ainsi que toute solution optimale. À aucun moment l'ordre des opérations imposé par le tableau de marqueurs n'est modifié et tout tableau de marqueurs peut être accédé depuis tout autre tableau de marqueurs (par applications successives de la mutation). Il en résulte que la condition d'accessibilité[5] est vérifiée par le couple $\langle$mutation, décodeur$\rangle$ puisqu'il existe au moins un chemin depuis n'importe quelle solution initiale admissible s_o vers une solution optimale s^*.

Mutations en codage direct Les opérateurs de mutation utilisés en codage direct sont des opérateurs « classiques » basés sur des permutations d'opérations critiques. Les propriétés de ces opérateurs ont déjà été étudiés par d'autres auteurs, aussi nous ne faisons que rappeler quelques propriétés relatives à ces opérateurs :

Propriété 9 (permutation d'opérations sur un chemin critique)
Un opérateur $\mathcal{O}$ qui permute deux opérations consécutives (sur une même machine) appartenant à un chemin critique vérifie la condition d'accessibilité [Tai93b, TCM95, HW96].

Propriété 10 (opérateurs et solutions réalisables)
Les solutions voisines $s' \in V_{\mathcal{O}}(s)$ obtenues par application de l'opérateur $\mathcal{O}$, défini précédemment, à une solution initiale réalisable s, sont réalisables.

Propriété 11 (opérateurs et bloc d'opérations critiques)
Soient $\mathcal{B}$ un bloc d'opérations critiques, et O_{ji} une opération appartenant à $\mathcal{B}$. Un opérateur $\mathcal{O}$ qui déplace O_{ji} en début ou en fin de $\mathcal{B}$ si la solution obtenue est réalisable ; ou dans une position proche du début ou de la fin de $\mathcal{B}$ si tel n'est pas le cas ; vérifie la condition d'accessibilité [HW96].

Crossover GA/GT Il n'est pas évident d'utiliser les notions d'ergodicité et d'accessibilité pour un crossover.

Une manière de définir l'ergodicité pour le crossover est la suivante : un crossover est ergodique s'il permet d'atteindre n'importe quel point (représenté par l'un des individus fils) de l'espace de recherche $\mathcal{E}$ à partir d'un couple de solutions initiales admissibles $\langle s_0, s_1\rangle$ en appliquant cet opérateur un certain nombre de fois à $\langle s_0, s_1\rangle$.

[3] voir définition 13, page 24
[4] voir définition 42, page 54
[5] voir définition 12, page 24

Une autre manière de définir l'ergodicité pour le crossover est la suivante : un crossover est ergodique s'il permet d'atteindre n'importe quel couple (représenté par ses individus fils) de l'espace de recherche $\mathcal{E}^2$ à partir d'un couple de solutions initiales admissibles $\langle s_0, s_1 \rangle \in \mathcal{E}^2$ en appliquant cet opérateur un certain nombre de fois à $\langle s_0, s_1 \rangle$.

Une manière de définir la condition d'accessibilité est la suivante : un crossover garantit la condition d'accessibilité s'il existe au moins un chemin depuis n'importe quel couple d'individus parents vers au moins un individu fils correspondant à une solution optimale.

Nous n'étudions pas davantage les propriétés du crossover GA/GT compte tenu de la complexité de cet opérateur.

8.2.3 Discussion

Lors de la définition d'un codage, il est nécessaire de considérer l'ensemble des composants qui vont interagir, c'est-à-dire les opérateurs et la fonction coût. Nous avons étudié plusieurs codages pour le *jobshop* simple. Nous considérons différents couples $\langle \text{codage}, \text{opérateur} \rangle$; l'objectif est d'utiliser un codage suffisamment riche pour définir des opérateurs efficaces. Les interactions avec la fonction coût sont plus délicates à mettre en évidence. Par exemple il peut s'agir de définir un codage et des opérateurs suffisamment élaborés pour véhiculer des informations complémentaires (critères secondaires) utilisables pour guider la recherche au travers de la fonction coût. Dans le cas du *jobshop*, il peut s'agir de véhiculer des informations relatives aux opérations critiques.

La notion de paysage adaptatif $\mathcal{P}$, définie par le triplet $\langle \mathcal{E}, \mathcal{O}, f \rangle$, illustre parfaitement cette interdépendance : changer d'opérateur ou de fonction objectif peut modifier complètement la nature du paysage et rendre de ce fait la recherche d'un optimum plus ou moins facile pour une méthode de recherche donnée.

8.3 Recherche tabou

Une fois définis les couples $\langle \text{codage}, \text{opérateur} \rangle$ retenus, nous les utilisons en premier lieu dans la recherche tabou. Nous précisons, dans cette section, la méthode de gestion de la liste tabou et les caractéristiques générales de la recherche tabou utilisée dans le cadre de nos recherches.

8.3.1 Liste tabou de taille variable

La variation de la taille de la liste tabou au cours du temps engendre une taille de voisinage variable en agissant sur le nombre de mouvements « autorisés » à un instant donné. Bien qu'utilisable en codage direct comme en codage indirect, nous avons expérimenté cette technique en codage direct. D'une part, en raison des avantages du codage direct (voir 8.2.1.1, page 137). D'autre part, cette recherche tabou est vouée à être intégrée à un algorithme génétique reposant sur un codage direct.

Étant donné que notre recherche tabou est destinée à être intégrée au sein de méthodes hybrides, nous avons opté pour une implantation extrêmement simplifiée de la recherche tabou (dépourvue de mécanisme de mémorisation à long terme, ...) non destinée à un fonctionnement autonome. De ce fait, nous ne détaillons pas les résultats de notre recherche tabou, utilisée de manière autonome, étant donné la simplicité de notre implantation face aux performances d'une recherche tabou « complète ». Inspirée d'une version simplifiée de la recherche tabou de [HW95], notre implantation s'apparente plus à un AIRL doté d'une liste tabou qu'à une recherche tabou en tant que telle.

Il faut cependant signaler que la recherche tabou à d'ores et déjà fait ses preuves dans de nombreux domaines. Cette méthode fait en général appel à des mécanismes complexes destinés par exemple à favoriser l'exploration de l'espace de recherche, ou encore à faciliter la recherche malgré la présence de nombreux optima locaux.

Pour clôturer cette section, nous mentionnons quelques exemples de recherches tabou performantes appliquées au domaine de l'ordonnancement [NS96b, TCM95, HW95, DG94, Özd96, EAVA97, HW96].

8.4 Algorithme Évolutif

Après avoir exposé les caractéristiques générales de la recherche tabou mise en œuvre, nous détaillons, dans cette section, les particularités de l'algorithme évolutif utilisé dans le cadre de nos recherches.

8.4.1 Méthodes de sélection

L'objectif des méthodes de sélection consiste à favoriser la reproduction des individus performants au sein de la population. Il existe de très nombreuses variantes de ces méthodes [Bak85, Bak87, Han94, Gol94, Mit96, RP96, Whi89, PM98]. Hormis la méthode dite « de sélection par roue de loterie[6] » qui présente de sérieux inconvénients dans sa version la plus simple, il n'existe pas de méthode de sélection dominante.

La méthode de sélection retenue doit éviter de provoquer l'apparition du *phénomène de super-individu* : supposons que la valeur moyenne des coûts des individus de la population obtenue à l'instant t soit égale à $\overline{f}$, par application des opérateurs génétiques un *super-individu* x^* extrêmement performant peut être engendré (c'est-à-dire que si l'objectif consiste à minimiser f, alors $f(x^*) \ll \overline{f}$, ou $f(x^*) \gg \overline{f}$ sinon). En l'absence de mécanisme particulier, la méthode de sélection peut favoriser x^* à tel point que la population à l'instant $t+1$ soit *inondée* de copies de l'individu x^*, ce qui risque de provoquer à court terme une convergence de la population.

Pour éviter ce phénomène, la sélection doit contrôler la *pression de sélection* de manière à maintenir un bon compromis entre la diversité de la population et la vitesse de convergence vers une solution de qualité. Nous définissons la pression de sélection de la manière suivante : la pression de sélection est le nombre de descendants qui seront attribués au meilleur individu x^* de la population à l'instant t, lors de la constitution de la nouvelle population $(t+1)$.

Par exemple, pour la méthode de la roulette, la pression de sélection est proportionnelle à $\frac{f(x^*)}{\overline{f}}$ (pour un problème de maximisation du coût).

Dans le cadre de nos travaux, nous utilisons une sélection de type « échantillonnage stochastique du reste » (SUS) sans doublon. Cette méthode de sélection permet de contrôler directement la pression de sélection via un paramètre appelé *biais* et présente de bonnes caractéristiques (mesurées expérimentalement par différents auteurs).

8.4.2 Impact des fonctions coût sur les performances d'un AE

De manière à déterminer si les résultats des fonctions multicritères[7] appliquées aux AIRL peuvent être généralisés aux algorithmes évolutifs, nous évaluons l'impact de l'ajout des critères secondaires

[6] encore appelée « méthode de la roulette »

[7] Par multicritère nous entendons bien entendu les fonctions multicritères telles que nous les avons définies et non les fonctions multicritères au sens large.

dans la fonction coût sur la qualité des solutions finales, sur la vitesse de convergence et sur la diversité de la population de l'AE. Dans notre cadre expérimental, la diversité est exclusivement mesurée en terme d'écart-type des coûts des individus. Une mesure complémentaire est effectuée lorsque des fonctions multicritères sont utilisées : la diversité est mesurée à la fois sur les coûts[8] des individus et sur les valeurs du makespan des individus. De cette manière nous étudions la variation de la diversité en terme de coût et de critère principal, c'est-à-dire en terme de makespan.

Comme pour les AIRL, nous utilisons les fonctions définies en section 7.3.2.4, page 106 sur le même ensemble d'instances de *jobshop*.

L'algorithme évolutif utilisé est une version modifiée de *LibGA* (voir section D.1, page 223). Il fonctionne en mode générationnel élitiste, les individus sont des ordonnancements (codage direct). Nous utilisons un opérateur de recombinaison basé sur le crossover GA/GT (voir section 8.2.2.1, page 138). La mutation effectue une permutation de deux opérations et ré-ordonnance les opérations lorsque cela est nécessaire (voir figure 7.2, page 82). Nous utilisons une sélection de type échantillonnage stochastique du reste (SUS sans doublon), plus précisément le paramétrage de l'AE est le suivant :

- codage : direct ;
- fonction coût : f_1, f_2, f_3 ou f_5 ;
- type d'algorithme évolutif : AG générationnel, élitiste ;
- taux de remplacement : 100% ;
- type de sélection : SUS sans doublon (biais de 1.25) ;
- crossover : GA/GT ;
- mutation : invert6 ;
- taux de crossover : 0.6 ;
- taux de mutation : 0.9 ;
- critère d'arrêt : nombre de générations ;
- taille de population : $2 \cdot J \cdot M$;
- nombre de générations : 2000.

Les résultats sont présentés dans le tableau 8.2. Les résultats en terme de makespan moyen sont conformes aux résultats de l'AIRLNDPM, c'est-à-dire que les fonctions multicritères dégradent les performances moyennes. Cependant, les résultats obtenus montrent que les meilleures performances (en terme de makespan minimum) sont systématiquement obtenues par les fonctions multicritères. Ceci s'explique par l'augmentation artificielle de la diversité induite par l'utilisation des critères secondaires. De ce fait, l'algorithme évolutif peut davantage progresser avant convergence de la population. Selon les fonctions coût utilisées, il peut en résulter une augmentation de l'écart-type et par conséquent une plus grande dispersion des valeurs de makespan obtenues en fin d'exécution.

Les résultats présentés en annexe A.3 confirment les résultats précédemment exposés. Ces résultats montrent également l'importance de la taille de population : bien que les fonctions multicritères augmentent artificiellement la diversité, l'augmentation de la taille de population permet d'améliorer les résultats. Les résultats présentés en annexe A.3 concernent plus particulièrement l'instance `ta01`, mais d'autres résultats obtenus lors de notre étude confirment cette tendance.

[8] Le coût d'un individu tient compte des éventuels critères secondaires. Il est obtenu par application de la fonction d'évaluation à l'individu considéré.

Instance	ft10	ft20	ta01
Taille	10x10	20x5	15x15
Optimum	930	1165	1231
f_1 Moyenne	988.3	(1208.5)	(1312.5)
Écart-type	19.5	21.9	17.2
Minimum	943 (1.4)	1178 (1.1)	1281 (4.1)
Maximum	1032	(1265)	(1352)
f_2 Moyenne	989.1	1210.9	1318.8
Écart-type	20.7	(20.6)	18.8
Minimum	947 (1.8)	(1173) (0.69)	1284 (4.3)
Maximum	1043	1273	1370
f_3 Moyenne	991.3	1212.2	1320.0
Écart-type	20.7	22.7	19.5
Minimum	(937) (0.75)	1178 (1.1)	1279 (3.9)
Maximum	1034	1269	1364
f_5 Moyenne	(988.1)	1208.8	1316.7
Écart-type	(16.7)	22.0	(15.6)
Minimum	946 (1.7)	1180 (1.3)	(1277) (3.7)
Maximum	(1021)	1279	1361

Tableau 8.2: Résultats de l'AE pour différentes fonctions coût. La deuxième ligne indique la taille de l'instance. La troisième ligne donne le makespan optimum, ou un encadrement lorsqu'il n'est pas connu. Ensuite, pour chaque fonction, le tableau contient la valeur moyenne et l'écart-type du makespan obtenus sur 60 exécutions. Le même ensemble de 60 populations initiales est utilisé pour les 4 fonctions coût afin de comparer précisément leurs performances relatives. Les résultats (*entourés*) correspondent aux valeurs minimales.

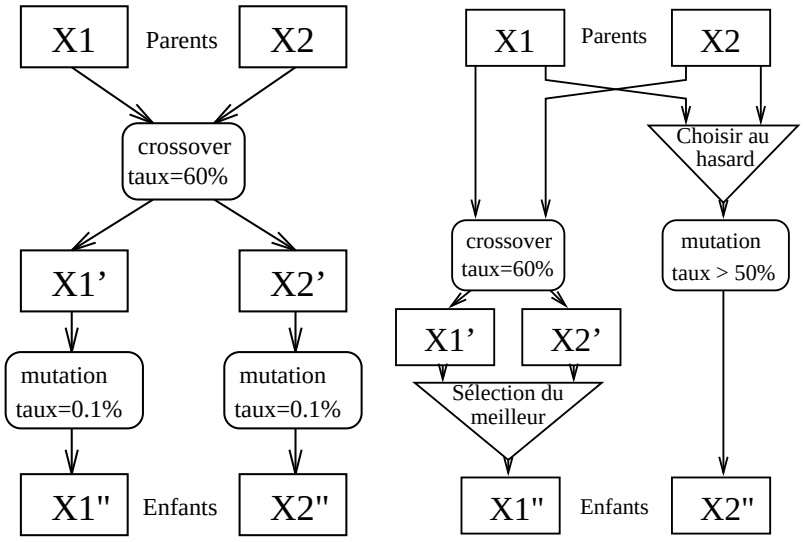
Les résultats montrent que les critères ajoutés permettent de modifier le comportement de l'algorithme évolutif dans l'espace de recherche de manière à aboutir à de meilleures solutions (en terme de meilleur individu découvert). Pour certaines instances, les fonctions multicritères permettent à la fois d'améliorer le meilleur individu obtenu, mais également les performances moyennes.

Les différences observées entre l'AE et l'AIRLNDPM sont dues aux interactions de la sélection, mais surtout du crossover avec les différentes fonctions coût. Les variations dans les performances obtenues sont vraisemblablement dues à l'opérateur de recombinaison GA/GT : dans sa version actuelle, cet opérateur optimise le makespan sans tenir compte des critères secondaires. La définition de nouveaux opérateurs intégrant ces critères devrait améliorer les performances obtenues.

8.4.3 Schéma d'application des opérateurs

Dans l'algorithme génétique standard, le schéma d'application des opérateurs est le suivant : deux individus sont sélectionnés dans la population ; l'opérateur de recombinaison (crossover) est appliqué, avec une certaine probabilité, sur ces deux individus. Il engendre deux individus sur lesquels l'opérateur de mutation est appliqué. Nous appelons ce schéma d'application, le schéma *séquentiel* (voir figure 8.4(a)).

Nous utilisons un schéma d'application des opérateurs différent de celui qui est traditionnellement utilisé : deux individus sont sélectionnés, ils engendrent un nouvel individu par crossover, et un individu par mutation. Contrairement au schéma traditionnel, la mutation n'est pas appliquée après le crossover (voir figure 8.4(b)). Nous appelons ce schéma d'application, le schéma *parallèle*.



(a) Schéma séquentiel (b) Schéma parallèle

Figure 8.4: Schémas d'application des opérateurs.

Il est ainsi possible de réduire les interactions entre crossover et mutation dues à l'utilisation d'opérateurs dédiés. En effet, les meilleurs résultats – en terme de qualité de solution – sont obtenus pour un taux de mutation élevé. En utilisant le schéma classique, nous avons constaté que le crossover était capable d'engendrer de très bonnes solutions. Cependant, l'opérateur de mutation détériore, en général, la qualité des solutions engendrées par le crossover. Ce schéma non traditionnel permet d'augmenter le taux de mutation tout en préservant les solutions engendrées par le crossover. L'utilisation d'un schéma parallèle ne se justifie que lorsque la mutation est susceptible d'engendrer

un individu-fils dont le coût est supérieur[9] à celui de l'individu-parent. Il peut également être utile de mettre en œuvre un schéma similaire lorsque l'opérateur de recombinaison engendre un seul individu (par combinaison linéaire des individus parents par exemple).

L'utilisation du schéma parallèle ne comporte pas que des avantages. Elle risque en effet d'engendrer une convergence prématurée de l'algorithme génétique de par la présence d'une étape de sélection entre $X'1$ et $X'2$ (voir figure 8.4). Pour limiter cet effet, l'étape de sélection peut être rendue non déterministe en utilisant des méthodes classiquement mises en œuvre dans l'étape de sélection des algorithmes génétiques. Une autre cause de convergence prématurée peut être énoncée de la manière suivante : un risque de convergence prématurée existe lorsque les individus engendrés par crossover sont de bien meilleure qualité que les individus engendrés par mutation. En effet, si les individus engendrés par mutation sont de très mauvaise qualité, il ont très peu de chance de survivre au fil des générations, il peut alors s'avérer utile de diminuer la pression de sélection au sein de l'étape de sélection de l'algorithme génétique.

Cette convergence rapide provoquée par le schéma parallèle n'est pas nécessairement néfaste : si l'algorithme génétique est utilisé au sein d'une méthode hybride, il peut alors s'avérer utile d'utiliser l'algorithme génétique pour converger rapidement vers une solution acceptable pour ensuite utiliser la solution obtenue par l'algorithme génétique comme solution de départ pour une autre méthode de recherche.

Nous allons étudier les méthodes hybrides dans la prochaine section, mais avant d'aborder ce sujet, nous présentons les résultats obtenus par les différents schémas d'application au sein d'un algorithme génétique (non hybride) après avoir évalué l'impact sur le nombre d'évaluations induit par l'utilisation des deux schémas.

8.4.3.1 Estimation du nombre d'évaluations

Nous avons défini deux formules permettant d'estimer le nombre d'appels à la fonction coût (voir le tableau 8.3). Ces formules permettent d'effectuer un grand nombre de comparaisons de l'AG aux autres méthodes de recherche sans nécessiter de lancer effectivement l'AG. Elle offre également la possibilité d'estimer la durée d'exécution de l'AG lorsque la durée de calcul de la fonction coût est supérieure à la durée de calcul des opérateurs de l'AG (mutation, crossover, sélection).

Schéma séquentiel	Schéma parallèle
$N_{es} = g\,p\,\left[t_c + (1 - t_c)\,t_m\right] + p$	$N_{ep} = g\,p\,\left[t_c + \frac{t_m}{2}\right] + p$

Tableau 8.3: Estimation du nombre d'évaluations à partir des paramètres suivants :
- p, le nombre d'individus dans la population (supposé pair) ;
- g, le nombre de générations de l'algorithme génétique ;
- t_m, le taux de mutation ;
- t_c, le taux de crossover.

Nous considérons un algorithme de type générationnel dont l'écart entre générations est fixé à 1 (*i.e.* toute la population est remplacée d'une génération sur l'autre). Il est cependant très facile d'adapter les calculs à un algorithme génétique stationnaire ou à un écart entre générations inférieur à 1.

[9] Nous supposons ici que l'objectif consiste à minimiser le coût

Détail des calculs Ces formules sont assez simples à obtenir à partir des schémas d'application des opérateurs :

- considérons un individu en schéma séquentiel, il sera évalué si l'opérateur de recombinaison lui est appliqué, ou si l'opérateur de mutation lui est appliqué, soit un taux global de $t_g = t_c + (1 - t_c)\, t_m$;
- considérons un individu en schéma parallèle, il sera évalué si l'opérateur de recombinaison lui est appliqué, ou si l'individu est sélectionné (avec une probabilité $\frac{1}{2}$) et l'opérateur de mutation lui est appliqué. En schéma parallèle, mutation et crossover sont deux événements indépendants ce qui donne un taux global de $t_g = t_c + \frac{t_m}{2}$.

Pour avoir une estimation du nombre d'évaluations, nous faisons l'hypothèse que ce taux global est affecté à chaque individu et à chaque génération (pour être plus précis, il faudrait considérer des paires d'individus), sans oublier l'évaluation complète (p évaluations) de la génération initiale, soit $N_e = p\, g\, t_g + p$.

Application numérique Le but de l'application numérique présentée dans le tableau 8.4 est double : d'une part, comparer les résultats obtenus aux estimations fournies par les formules ; d'autre part, établir une première comparaison entre schéma séquentiel et schéma parallèle du point de vue du nombre d'évaluations. Diverses mesures ont été effectuées avec des résultats comparables, nous présentons ici les mesures obtenues en considérant le paramétrage utilisé pour les résultats présentés en annexe A.3.

La différence entre le nombre d'évaluations estimées et le nombre d'évaluations calculées est inférieure à 0.2%, ce qui est largement suffisant pour notre étude. Cependant, en analysant les résultats, nous constatons que la valeur obtenue par calcul est systématiquement sous-estimée d'environ p évaluations (où p est la taille de population). Cette différence est due au fonctionnement interne de l'AG utilisé (*LibGA*) qui nécessite p évaluations supplémentaires pour le calcul de statistiques sur l'expérience courante. En tenant compte de cette correction, la différence passe de moins de 0.2% à moins de 0.15‰, mais il n'est plus possible de déterminer si la valeur calculée sur-estime ou sous-estime le nombre effectif d'appels à la fonction coût.

Pour être plus précis, les formules utilisées montrent que pour un taux de crossover supérieur à 0.5, le schéma parallèle est plus coûteux (en terme de nombre d'appels à la fonction coût) que le schéma séquentiel, sauf pour le cas où le taux de mutation est fixé à zéro (voir figure 8.5).

De manière générale, le schéma parallèle est utilisé pour des taux de recombinaison supérieurs à 0.5 et de forts taux de mutation. Or, cela correspond au domaine (voir figure 8.5) où le schéma séquentiel est moins coûteux que le schéma parallèle. Par conséquent, nous allons effectuer une série d'expériences pour évaluer les performances relatives (en terme de coût moyen de solution finale) des deux schémas.

8.4.3.2 Résultats comparatifs

Dans un premier temps, nous utilisons le même paramétrage pour les deux schémas d'application testés au sein de l'algorithme génétique et nous mesurons les performances obtenues après une génération ainsi qu'en fin d'exécution de l'algorithme génétique. Pour tenir compte du fait que le taux d'application de l'opérateur de mutation n'a pas la même signification dans les deux schémas d'application, nous avons testé plusieurs paramétrages en faisant varier le taux d'application de l'opérateur de mutation. Nous ne présentons dans cette section que les résultats les plus significatifs.

paramètres			Taille de population = 500, 300 générations				
Taux		Schéma	Estimé	Mesuré			
t_m	t_c	d'application		moy	écart	max	min
0.9	0.6	séquentiel	144500	145029	73.6	145137	144913
0.9	0.6	parallèle	158000	158384	326.6	158975	157681
0.9	0.9	séquentiel	149000	149500	30.4	149542	149453
0.9	0.9	parallèle	203000	203472	180.8	203751	203191
paramètres			Taille de population = 900, 1000 générations				
Taux		Schéma	Estimé	Mesuré			
t_m	t_c	d'application		moy	écart	max	min
0.9	0.6	séquentiel	864900	865817	195.1	866117	865469
0.9	0.6	parallèle	945900	947081	695.5	948370	946169
0.9	0.9	séquentiel	891900	892811	125.1	892985	892552
0.9	0.9	parallèle	1215900	1216978	341.8	1217700	1216326
paramètres			Taille de population = 1600, 1000 générations				
Taux		Schéma	Estimé	Mesuré			
t_m	t_c	d'application		moy	écart	max	min
0.9	0.6	séquentiel	1537600	1539259	203.6	1539482	1538825
0.9	0.6	parallèle	1681600	1683180	841.9	1685146	1682275
0.9	0.9	séquentiel	1585600	1587210	140.0	1587439	1586994
0.9	0.9	parallèle	2161600	2163201	715.7	2164581	2162285

Tableau 8.4: Comparaison du nombre d'évaluations estimé au nombre d'évaluations mesuré au niveau de l'algorithme génétique. Pour chaque paramétrage de l'AG, dix expériences sont utilisées pour calculer les statistiques présentées ci-dessus. Nous choisissons les valeurs de taux de mutation (t_m), de taux crossover (t_c) et de taille de population (p) conformément aux résultats présentés en annexe A.3.

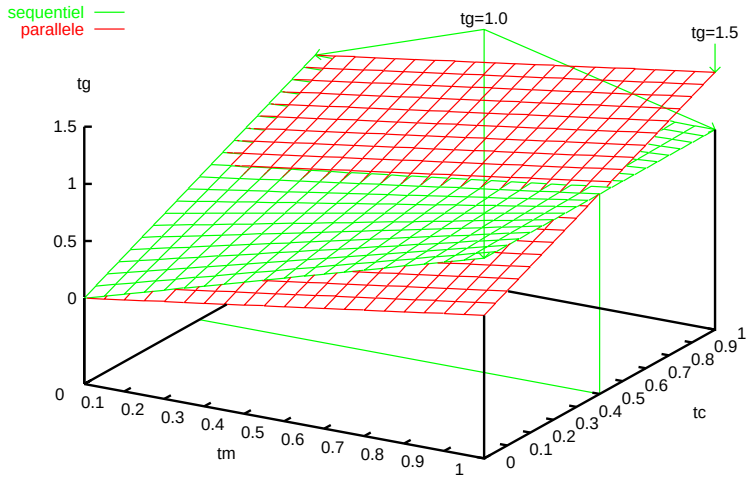


Figure 8.5: Comparaison du taux global d'application des opérateurs (t_g) pour les deux schémas : t_g est assimilable au nombre moyen d'évaluations d'un individu par génération, plus il est élevé, plus le nombre d'évaluations est important (et inversement).

Le paramétrage varie selon les expériences effectuées (voir annexe A.3, page 180). La sélection effectuée à l'issue du crossover dans le schéma d'application parallèle est déterministe : elle choisit systématiquement le meilleur individu.

Nous comparons les deux schémas d'application des opérateurs en utilisant pour chacun d'entre eux le paramétrage ayant donné les meilleurs résultats sur les instances considérées.

Les résultats présentés en annexe A.3 montrent que le schéma d'application parallèle donne systématiquement les meilleurs résultats lorsque la fonction coût exclusivement basée sur le makespan (f_1) est utilisée.

Critère étudié	Inst	f_1		f_2		f_3	
		sch	$C_{\max}$	sch	$C_{\max}$	sch	$C_{\max}$
moy($\Pi'_{\min}$)	ft10	par	975.7	séq	974.2	séq	972.9
moy($\Pi'_{\min}$)	ft20	par	1215.8	par	1212.2	par	1212.2
moy($\Pi'_{\min}$)	ta01	par	1340.3	par	1333.7	par	1341.5
moy($\Pi'_{\min}$)	abz7	par	728.5	par	725.3	par	724.5
moy(Π'_{moy})	ft10	par	976.6	séq	977.3	séq	975.2
moy(Π'_{moy})	ft20	par	1218.0	par	1213.5	par	1215.4
moy(Π'_{moy})	ta01	par	1342.0	par	1335.1	par	1342.4
moy(Π'_{moy})	abz7	par	730.1	par	726.6	par	725.4

Tableau 8.5: Résultats complémentaires de l'AE pour différentes fonctions coût. Le premier critère correspond à la moyenne des meilleurs individus obtenus sur chacune des dix expériences lancées. Le second critère étudié correspond à la moyenne sur les expériences de la moyenne des coûts des individus de la population finale pour chacune des dix expériences. Le nombre de générations est fixé à 300. La population comporte 500 individus. La colonne sch donne le nom du schéma d'application utilisé (séquentiel ou parallèle).

Critère étudié	f_1		f_2		f_3	
	sch	$C_{\max}$	sch	$C_{\max}$	sch	$C_{\max}$
moy($\Pi'_{\min}$)	par	1312	séq	1305	séq	1310
moy(Π'_{moy})	par	1313	séq	1306	séq	1310

Tableau 8.6: Résultats de l'AE pour différentes fonctions coût sur l'instance ta01. Le premier critère correspond à la moyenne des meilleurs individus obtenus sur chacune des dix expériences lancées. Le second critère étudié correspond à la moyenne sur les expériences de la moyenne des coûts des individus de la population finale pour chacune des dix expériences. Le nombre de générations est fixé à 1000. La population comporte 1600 individus. La colonne sch donne le nom du schéma d'application utilisé (séquentiel ou parallèle).

Cependant, en fonction du paramétrage, les résultats varient lorsque les fonctions multicritères sont utilisées. Pour illustrer notre propos, le tableau 8.5 synthétise les résultats relatifs au paramétrage constitué de 300 générations de 500 individus présentés en annexe A.3. Pour ce paramétrage les meilleurs résultats sont obtenus lorsque le schéma parallèle et les fonctions multicritères sont utilisés (à l'exception de l'instance ft10). En revanche, si nous étudions les résultats issus de l'annexe A.3.12 (page 208), le tableau 8.6 montre que dès qu'une fonction multicritère est utilisée le schéma séquentiel donne les meilleurs résultats.

Une fois encore l'opérateur de recombinaison GA/GT semble être la cause de ces résultats : nous avons étudié (section A.3.12, page 208) l'impact de l'opérateur de mutation interne du crossover GA/GT. Une fois activée, les résultats sont complètement inversés et les meilleurs résultats sont, non seulement obtenus lorsque la fonction multicritère f_3 est utilisée ; mais il apparaît également que le paramétrage correspondant à un taux d'application maximum (soit 0.9) de l'opérateur GA/GT donne les meilleurs résultats.

8.4.4 Discussion

Étant donné que nous utilisons typiquement un taux de mutation élevé, et un opérateur de recombinaison complexe, il arrive fréquemment que le bénéfice d'une application du crossover soit détruit par l'application de la mutation effectuée juste après. Aussi, nous utilisons un schéma différent d'application des opérateurs dans lequel le crossover et la mutation sont appliqués de manière exclusive et non plus en séquence. Ce schéma nous permet d'améliorer les performances obtenues. Appliqué dans une méthode hybride, nous pouvons utiliser ce schéma pour accélérer la convergence de l'algorithme vers des solutions de bonne qualité. Les résultats présentés utilisent une sélection déterministe en sortie de l'opérateur de recombinaison et une sélection non déterministe pour choisir l'individu parent qui subira la mutation. Cependant, il est tout a fait envisageable de remplacer ces sélections « élémentaires » par d'autres méthodes plus évoluées afin de mieux contrôler le rapport entre vitesse de convergence et qualité de solution finale. Clairement, l'objectif visé dans cette section consiste à montrer que nous pouvons également améliorer la qualité des solutions obtenues par l'AE, plutôt que de paramétrer l'AE de manière à obtenir les meilleurs résultats.

Nous avons illustré grâce à différents paramétrages, l'impact – sur les performances de l'AE – de la taille de population, du taux d'application des opérateurs, de l'utilisation de différentes fonctions coût et enfin de la modification de l'opérateur de recombinaison.

Dans la prochaine section, nous étudions l'impact de ces paramètres dans deux méthodes hybrides.

8.5 Méthodes hybrides

Cette section est consacrée à l'étude de méthodes hybrides constituées à partir d'un algorithme évolutif, d'un AIRLNDP et d'une recherche tabou dans leur version *quasi-canonique*.

8.5.1 Algorithme génétique hybride

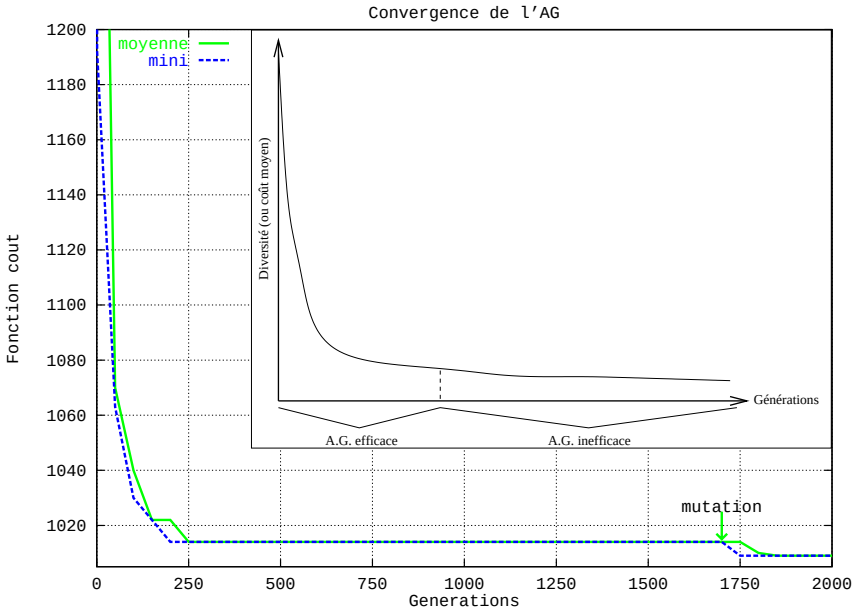


Figure 8.6: Convergence d'un AG lors de la résolution du *jobshop*.

L'un des problèmes rencontrés lors de l'application des AG au *jobshop* est la perte de diversité très rapide de la population (voir figure 8.6). De manière à pallier ce problème, nous avons conçu deux algorithmes génétiques hybrides.

① Nous avons tout d'abord réalisé un algorithme génétique hybride basé sur un AG et un AIRL. Nous utilisons un schéma d'hybridation de type « parallèle synchrone » [PT95b] : l'AIRL est utilisé par l'AG à la manière d'un opérateur de mutation (voir figure 8.7).

② Ensuite, un algorithme génétique hybride utilisant la recherche tabou a également été réalisé afin d'améliorer les performances obtenues (voir figure 8.8). Cet algorithme hybride tire profit de la complémentarité des méthodes hybridées : l'algorithme génétique possède une connaissance « globale » au travers de sa population et explore l'espace de recherche, tandis que la recherche tabou détient une connaissance locale et exploite le voisinage.

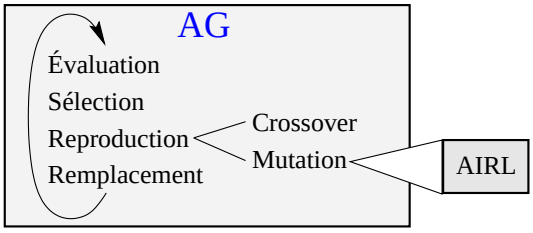


Figure 8.7: Hybridation parallèle synchrone : un opérateur (ici la mutation) est remplacé par une méthode de recherche (ici un AIRL). L'algorithme hybride résultant est désigné par la notation AG[AIRL].

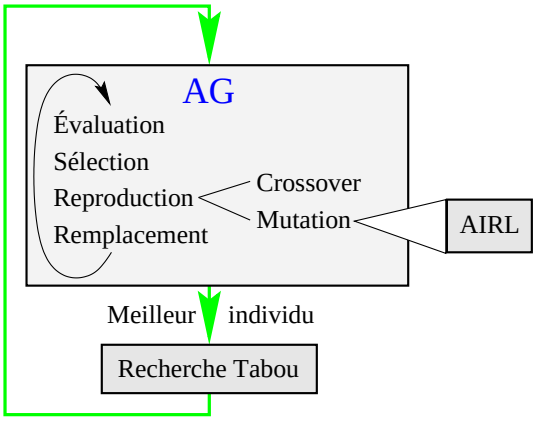


Figure 8.8: Algorithme génétique hybride AG[AIRL]+Tabou. Cet algorithme est constitué d'une double hybridation : une hybridation parallèle synchrone avec un AIRL et une hybridation séquentielle avec une recherche tabou. L'AIRL est suffisamment rapide pour être utilisé comme opérateur de mutation au prix d'une faible augmentation de la durée d'exécution. En revanche, la recherche Tabou, plus lente, n'est appliquée qu'au meilleur individu de la population.

8.5.1.1 Résultats

Dans cette section, nous étudions les performances des modèles hybrides, et plus particulièrement l'influence de l'hybridation sur la qualité des solutions. Les résultats concernant les méthodes hybrides ont fait l'objet de plusieurs publications et rapports internes [DPT95c, DPT95a,

DPT96a, DPT96b, DPT95b, DPT95d]. Le tableau 8.7 présente une synthèse de ces résultats sur quelques instances de la *OR-Library*. Chaque colonne correspond à une méthode heuristique : AG est un algorithme génétique, AG[] est un algorithme hybride constitué d'un AG et d'un AIRLNDP, AG[]+Tabou est un algorithme hybride constitué de AG[] et d'une recherche tabou, et Tabou est une recherche tabou (dans une version quasi-canonique). Pour l'hybride AG[]+Tabou, la recherche tabou est appelée toutes les 10 générations.

Les meilleurs résultats obtenus par un AIRL itéré (MSAIRL) depuis le début de nos travaux sont rassemblés dans le tableau 8.8. Ces résultats permettent d'une part de vérifier la condition d'accessibilité d'une solution optimale pour nos opérateurs (lorsqu'une solution optimale est découverte par le MSAIRL). D'autre part, ils donnent en quelque sorte une borne inférieure pour les hybrides afin de s'assurer que pour la grande majorité des instances (aux « coups de chances » près) l'algorithme génétique hybride fournit au moins des solutions de qualité équivalente avec une probabilité plus importante.

Les résultats montrent que les hybrides fournissent de meilleurs résultats par rapport à l'AG non-hybride. Les résultats coïncident avec ceux de nombreux auteurs qui ont constaté l'amélioration des performances lors de l'hybridation des algorithmes génétiques avec d'autres méta-heuristiques ou des heuristiques dédiées au *jobshop*. Notre algorithme génétique hybride engendre des solutions optimales ou quasi-optimales sur une large gamme d'instances de tailles très variées.

Les résultats détaillés d'une série d'expériences sont donnés à l'annexe A.4, page 212

Instance	Taille	Optimum	AG[]+Tabou	Tabou	AG[]	AG
ft06	6x6	55	55 (0)	55 (0)	55 (0)	55 (0)
ft10	10x10	930	930 (0)	930 (0)	930 (0)	953 (2)
ft20	20x5	1165	1175 (<1)	1165 (0)	1178 (1)	1180 (1)
car1	11x5	7038	7038 (0)	7038 (0)	7038 (0)	7101 (<1)
car3	12x5	7312	7312 (0)	7399 (1)	7312 (0)	7594 (4)
car5	10x6	7702	7702 (0)	7720 (≈0)	7702 (0)	8047 (4)
car8	8x8	8264	8294 (≈0)	8294 (≈0)	8264 (0)	8407 (2)
abz7	20x15	656	672 (2)	674 (<3)	685 (4)	757 (15)
la36	15x15	1268	1291 (<2)	1291 (<2)	1292 (<2)	1408 (11)
ta21	20x20	1539/1645	1692 ($\frac{\leq 10}{\geq 3}$)	1712 (≥4)	1721 (≥5)	1968 (≥20)
ta31	30x15	1764	1771 (≈0)	1774 (<1)	1907 (8)	2098 (19)
ta41	30x20	1859/2023	2108 ($\frac{\leq 14}{\geq 4}$)	2117 (≥5)	2262 (≥12)	2551 (≥26)
ta51	50x15	2760	2763 (≈0)	2776 (<1)	2909 (5)	3225 (17)
ta71	100x20	5464	5490 (≈0)	5609 (<3)	5723 (<5)	6065 (9)

Meilleure solution : AG[]+Tabou > Tabou > AG[] > AG

Légende :
- $AG[]$ signifie $AG[AIRLNDP]$
- ≈0 représente un résultat < 0.5% de l'optimum
- <1 représente un résultat ≥ 0.5% de l'optimum (et < 1%)

Tableau 8.7: Résultats de différentes heuristiques : bien que les méta-heuristiques (AIRLND, Tabou, AG) utilisées pour constituer les algorithmes présentés soient des versions quasi-canoniques, les résultats obtenus par les hybrides sont de bonne qualité. Pour les AG et les AG hybrides, dix expériences sont lancées pour chaque instance. Le résultat affiché dans le tableau correspond à la meilleure valeur de makespan obtenue.

Instance	Taille	Optimum	MSAIRL
ft6	6x6	55	55 (0)
ft10	10x10	930	930 (0)
ft20	20x5	1165	1175 (<1)
car1	11x5	7038	7038 (0)
car3	12x5	7312	7312 (0)
car5	10x6	7702	7702 (0)
car8	8x8	8264	8264 (0)
abz7	20x15	656	682 (4)
la36	15x15	1268	1276 (<1)
ta21	20x20	1539/1645	1717 (≥4)
ta31	30x15	1764	1807 (2)
ta41	30x20	1859/2023	2143 (≥6)
ta51	50x15	2760	2760 (0)
ta71	100x20	5464	5744 (5)

Tableau 8.8: Meilleurs résultats de l'AIRL itéré (*Multi-Start* AIRL) : lors de la mise au point de nos opérateurs et de nos algorithmes, de nombreux benchmarks préliminaires à notre étude ont été lancés. Bien que non présentés dans ce mémoire, il nous a semblé opportun de tenir à jour une liste des meilleurs résultats obtenus par l'AIRL itéré. Les résultats montrent que l'AIRL itéré obtient des résultats excellents (surtout si l'on tient compte de l'extrême simplicité de l'algorithme) sur les instances de petite ou moyenne taille, les résultats sont moins évidents sur des instances de plus grandes tailles.

8.6 Comparaison AG, AIRLD, et AIRLND

L'objectif de cette section est de comparer les performances de ces algorithmes, non seulement en terme de qualité de solution finale, mais surtout en terme de ressources nécessaires du point de vue du nombre d'évaluations et de mémoire occupée.

8.6.1 Évolution du coût des algorithmes

Les courbes présentées à la figure 8.9 permettent d'évaluer l'évolution du coût des algorithmes. Nous nous sommes concentrés sur quatre ou cinq instances pour illustrer notre démarche. Ces résultats ont été confirmés par de nombreux résultats obtenus sur les instances de la *OR-Library*.

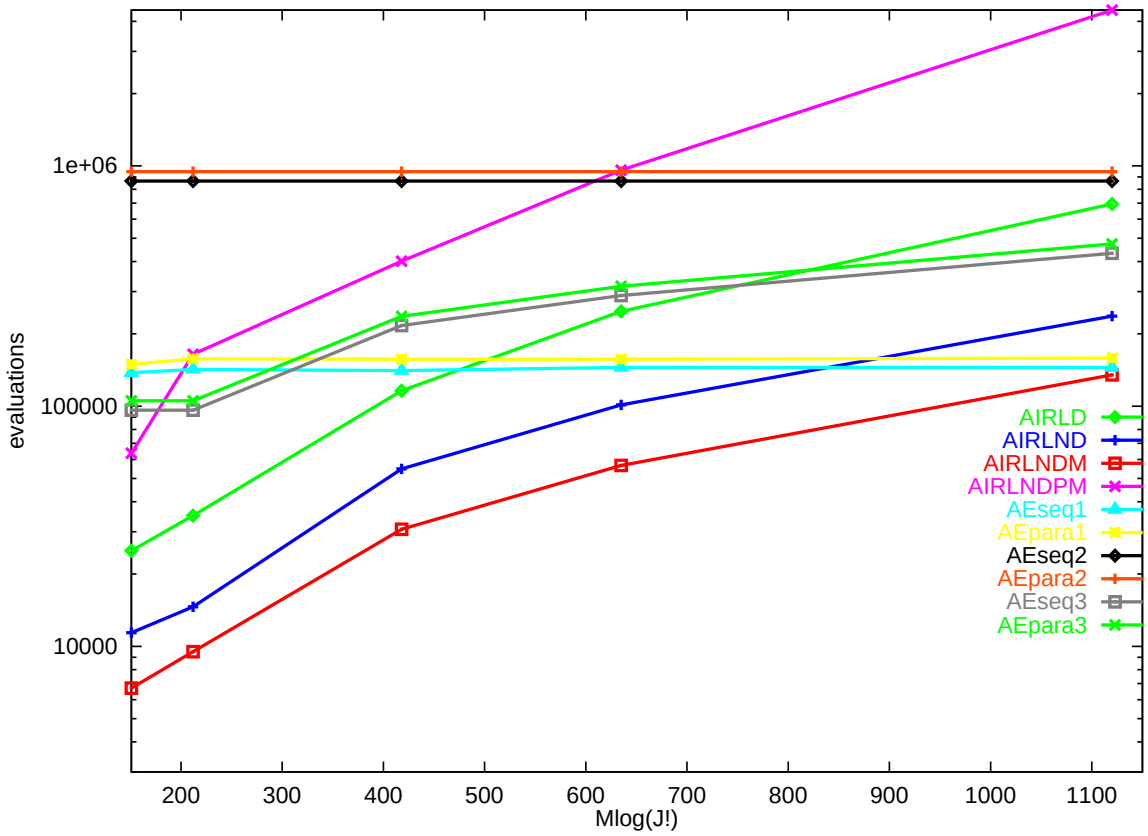
Les courbes permettent de distinguer trois familles d'algorithmes : l'algorithme évolutif, l'AIRLNDPM et les AIRL non dotés d'un mécanisme de déplacement sur les plateaux. Pour ces derniers, les résultats étaient prévisibles : bien que le nombre d'évaluations augmente avec la taille des instances, la qualité des solutions finales se détériore.

Pour l'AIRLNDPM, les performances sont meilleures que les autres AIRL mais au prix d'un très grand nombre d'évaluations (malgré l'utilisation d'une échelle logarithmique pour le nombre d'évaluations, les courbes permettent de séparer clairement l'AIRLNDPM des autres algorithmes).

Les courbes indiquent que l'AE constitue un bon compromis entre l'AIRLNDPM et les autres AIRL, autrement dit entre qualité de solution finale et nombre d'évaluations. Le paramétrage utilisé pour l'algorithme évolutif explique en partie ces résultats. En effet, la condition d'arrêt utilisée pour nos expériences est exclusivement basée sur le nombre de générations, ce qui induit un nombre d'évaluations constant lorsque la taille de l'instance augmente. Le paramétrage retenu offre un bon compromis entre nombre d'évaluations et qualité de solution finale pour des instances de taille inférieure à 30x15. Au delà de cette taille, il est souhaitable de revoir à la hausse la taille de la

Instance			AIRLD		AIRLND		AIRLNDM		AIRLNDPM		AEseq1		AEpara1	
Nom	$M\log(J!)$	JxM	éval	$\%_{opt}$	éval	$\%_{opt}$	éval	$\%_{opt}$	éval	$\%_{opt}$	éval	$\%_{opt}$	éval	$\%_{opt}$
ft10	151	10x10	25000	16.67	11400	16.56	6700	16.56	63700	6.13	137800	5.91	149050	6.13
ft20	212	20x5	35000	15.97	14600	16.14	9500	15.79	164500	3.52	142120	4.38	157450	4.64
ta01	418	15x15	116000	19.17	54800	18.60	30700	18.60	400900	4.22	140680	9.67	156925	9.18
abz7	635	20x15	248000	19.21	101200	19.21	56600	19.05	959900	5.64	145000	12.35	156925	11.74
ta31	1120	30x15	695000	21.88	237100	21.88	134900	21.88	4452400	4.48	145000	–	158500	–

(a) Relation entre la qualité de solution finale et le coût



(b) Relation entre le logarithme de la taille de l'espace et le nombre d'évaluations

Figure 8.9: Le tableau 8.9(a) présente l'évolution du coût (en nombre d'évaluations) en fonction de la qualité moyenne des solutions finales exprimée en pourcentage par rapport au makespan optimum (f_1). Nous faisons une comparaison en parallèle avec l'évolution du nombre d'évaluations en fonction du logarithme de la taille de l'espace de recherche sur les courbes 8.9(b). Les courbes intitulées AIRLD, AIRLND, AIRLNDM et AIRLNDPM correspondent aux performances moyennes des AIRL susnommés sur 500 expériences. Les courbes AEpara1 et AEseq1 sont issues des résultats exposés à l'annexe A.3.4 (moyenne du critère Π'_{min}). Elles correspondent respectivement à l'utilisation d'un schéma d'application parallèle ou séquentiel, pendant 300 générations sur une population de 500 individus avec un taux de mutation (invert1) de 0.9 et un taux de crossover de 0.6. Le même paramétrage des opérateurs est utilisé pour les courbes AEpara2 et AEseq2, issus des résultats exposés à l'annexe A.3.10, cependant l'AE est exécuté pendant 1000 générations de 900 individus. Les courbes AEpara3 et AEseq3 sont des estimations (réalisées grâce aux formules présentées en section 8.4.3.1) du nombre d'évaluations pour un AE effectuant 500 générations sur une population de taille $2{\cdot}J{\cdot}M$.
Note : le nombre d'évaluations nécessaires à l'obtention des solutions finales est borné par le critère d'arrêt utilisé pour l'AE. En effet, le critère d'arrêt n'est pas la convergence mais le nombre de générations.

population et le nombre de générations.

Il est à noter que la taille de population est sous-estimée, ce qui a tendance à provoquer des convergences prématurées de l'AE surtout lors de l'utilisation d'un schéma d'application parallèle. Pour s'en convaincre, il suffit de comparer les valeurs moyennes du critère $\Pi'_{\text{écart}}$ pour les schémas séquentiel et parallèle (annexe A.3.4, page 192). Malgré ce « handicap », le schéma parallèle donne de meilleurs résultats que le schéma séquentiel pour les instances ta01 et abz7.

Compte tenu du nombre de paramètres de l'AE, il est possible de le paramétrer de manière à obtenir une évolution du nombre d'évaluations similaire aux AIRL afin de comparer les algorithmes après un même nombre d'évaluations par exemple.

Pour certaines expériences, dont l'objectif est d'obtenir rapidement des solutions sans nécessairement chercher à laisser converger l'AE, nous utilisons de manière empirique une taille de population égale à $2{\cdot}J{\cdot}M$. L'AE évolue pendant 250 à 500 générations selon le fait d'utiliser l'AE seul ou au sein d'une méthode hybride. Nous avons tracé les courbes relatives à 500 générations. Ces dernières montrent qu'avec un tel paramétrage, l'AE effectue plus d'évaluations que les AIRL pour les instances de petite taille alors que la tendance s'inverse pour les grandes tailles d'instances.

De nombreux auteurs se sont penchés sur le paramétrage des AE, parmi lesquels [NDY94] ont étudié le crossover GA/GT (voir section 8.2.2.1, page 138). Les résultats sont fortement dépendants du problème traité (au travers de la fonction coût), du codage, des opérateurs, des schémas d'application des opérateurs et du type d'AE (générationnel, stationnaire, hybride, etc.).

Fort heureusement, les algorithmes évolutifs, comme de nombreuses autres méta-heuristiques, offrent une certaine robustesse vis-à-vis de leur paramétrage : à la condition de rester dans une gamme de valeurs raisonnables, les résultats obtenus sont de très bonne qualité. Nous ne remettons pas en cause pour autant les études menées sur un problème particulier visant à déterminer le paramétrage *optimum* pour un problème précis. Nous précisons simplement en ce point que tel n'est pas notre objectif.

Au risque d'extrapoler les résultats de la figure 8.9, il semble raisonnable de penser que les courbes reflètent – d'une certaine manière – l'évolution de la qualité des solutions finales en fonction de l'effort de programmation (autrement dit, le temps passé à écrire le programme) : les AIRL non dotés de mécanisme de déplacement sur les plateaux sont extrêmement simples à implanter, très rapides mais peu performants surtout pour les grandes tailles d'instances. Les AIRL dotés d'un mécanisme élémentaire de déplacement sur les plateaux améliorent les performances mais sont vites pénalisés, du fait de l'absence de mécanismes élaborés, par le nombre d'évaluations. Finalement l'ajout de mécanismes plus sophistiqués (tels que les listes tabou, les mécanismes de partage d'information, de coévolution, ...) donne le meilleur compromis entre difficulté d'écriture du programme, qualité de solution finale et vitesse d'exécution (ou indirectement nombre d'évaluations).

8.6.2 Nombre d'évaluations

Le codage utilisé par l'AIRLD et les AIRLND est une pseudo-permutation de marqueurs de taille $J{\times}M$, constituée de M marqueurs identiques pour chacun des J produits. Parmi les $P = \frac{JM(JM-1)}{2}$ permutations possibles, seules $N = \frac{M^2J(J-1)}{2}$ sont différentes. Le codage est redondant et $P - N$, soit $\frac{JM(M-1)}{2}$ configurations sont inutiles. L'opérateur utilisé engendre un voisinage de taille N. La taille du voisinage de quelques instances est donnée dans le tableau 8.9.

Le choix de ce codage et de cet opérateur n'est pas optimal : le but ici n'est pas de trouver l'optimum, mais de mettre en évidence les interactions entre les fonctions coût et les déplacements sur les plateaux, dans un paysage où existe un grand nombre de plateaux. Nous pouvons ainsi comparer le comportement de deux familles d'AIRL simples :

Instance	ft10	ft20	la29	ta01	abz7	la31	ta21	ta31	ta41	ta51	ta61	ta71
JxM	10x10	20x5	20x10	15x15	20x15	30x10	20x20	30x15	30x20	50x15	50x20	100x20
Voisinage	4500	4750	19000	23625	42750	43500	76000	97875	174000	275625	490000	1980000

Tableau 8.9: Taille du voisinage pour l'AIRLD.

- les AIRL capables de se déplacer sur les plateaux ;
- les AIRL non autorisés à se déplacer sur les plateaux.

L'impact de la « précision » et du « bruitage » de la fonction coût est loin d'être négligeable ...

Définition 62 (précision de la fonction coût)
Soient f une fonction coût et $\mathcal{C}$ le cardinal du plus grand ensemble de solutions différentes ayant la même valeur de coût pour f. La précision de la fonction coût vaut :

$$P(f) = \frac{1}{\mathcal{C}}$$

Plus $P(f)$ est proche de 1, plus f est précise.

Pour le problème étudié (le *jobshop* simple), nous avons la possibilité de réduire la taille du voisinage sans perdre la condition d'accessibilité à un optimum global : il suffit pour cela de considérer les permutations d'opérations critiques. Il en résulte une diminution de la taille des plateaux et une augmentation de la précision de la fonction coût. Nous pouvons alors étudier les différents algorithmes dans ce « nouveau » paysage.

Nous utilisons un AIRLD comme algorithme de référence pour comparer différents algorithmes itératifs, en terme de qualité de la solution finale et de nombre d'évaluations.

À chaque pas, l'AIRLD effectue N évaluations ; le nombre moyen de pas et d'évaluations pour les instances utilisées sont indiqués dans le tableau 8.10.

Instance		ft10	ft20	ta01	abz7	ta31
JxM		10x10	20x5	15x15	20x15	30x15
f_0	Pas	8.7	10.8	10.0	11.0	16.6
	Éval	390	510	2360	4700	16250
f_1	Pas	5.5	7.4	4.9	5.8	7.1
	Éval	250	350	1160	2480	6950
f_2	Pas	8.9	13.9	12.9	13.5	24.5
	Éval	400	660	3050	5770	23980
f_3	Pas	9.1	15.0	13.2	14.1	25.8
	Éval	410	710	3120	6030	25250
f_4	Pas	7.5	9.6	7.2	7.0	8.5
	Éval	340	460	1700	2990	8320
f_5	Pas	6.1	8.7	5.5	7.4	8.6
	Éval	270	410	1300	3160	8420
f_6	Pas	9.6	28.2	12.9	17.2	32.8
	Éval	430	1340	3050	7350	32100

Tableau 8.10: Nombre moyen de pas et d'évaluations pour l'AIRLD. Le nombre d'évaluations est divisé par 100.

Pour l'AIRL non déterministe (AIRLND), l'AIRL non déterministe utilisant une mémoire de mouvements (AIRLNDM), ainsi que l'AIRL non déterministe utilisant une mémoire de mouve-

ments autorisé à évoluer sur les plateaux de f (AIRLNDPM), nous avons mesuré le nombre moyen d'évaluations pour les cinq instances utilisées (voir tableau 8.11).

Instance	ft 10	ft 20	ta 01	abz 7	ta 31
JxM	10x10	20x5	15x15	20x15	30x15
f_1 AIRLND	114	146	548	1012	2371
AIRLNDM	67	95	307	566	1349
AIRLNDPM	637	1645	4009	9599	44524
f_2 AIRLND	126	183	589	1058	2642
AIRLNDM	78	127	338	605	1524
AIRLNDPM	732	3273	4884	12027	78671
f_3 AIRLND	127	185	591	1068	2687
AIRLNDM	78	136	342	620	1545
AIRLNDPM	732	3236	4749	12357	77881

Tableau 8.11: Nombre moyen d'évaluations (divisé par 100) pour les différents AIRL non déterministes.

	AIRLND	AIRLNDM	AIRLNDPM
f_1	2.42	4.15	0.2618 ($\approx \frac{1}{3.82}$)
f_2	5.30	8.82	0.4314 ($\approx \frac{1}{2.32}$)
f_3	5.48	9.13	0.4497 ($\approx \frac{1}{2.22}$)

Tableau 8.12: Comparaisons du nombre moyen d'évaluations (sur les cinq instances étudiées) pour les AIRL : le tableau rassemble les ratio constitués à partir du nombre d'évaluations de l'AIRLD divisé par le nombre d'évaluations des différents AIRLND.

Pour les AIRL non déterministes (AIRLND, AIRLNDM, AIRLNDPM), nous avons mesuré le nombre moyen de pas pour les cinq instances utilisées (voir tableau 8.13). Les comparaisons détaillées des différents AIRL ont été effectuées dans les sections précédentes, le tableau 8.12 rassemble les comparaisons du point de vue du nombre d'évaluations.

Instance	ft 10	ft 20	ta 01	abz 7	ta 31
JxM	10x10	20x5	15x15	20x15	30x15
f_1 AIRLND	16(6)	24(8)	15(6)	18(6)	25(8)
AIRLNDM	30(12)	60(25)	53(23)	52(20)	129(53)
AIRLNDPM	29(7)	57(12)	49(9)	55(9)	126(21)
f_2 AIRLND	30(12)	60(25)	53(23)	52(20)	129(53)
AIRLNDM	30(11)	61(26)	53(22)	52(20)	129(53)
AIRLNDPM	55(16)	165(41)	164(45)	166(37)	727(198)
f_3 AIRLND	31(12)	63(25)	55(23)	54(21)	135(59)
AIRLNDM	31(11)	66(28)	55(22)	54(20)	134(54)
AIRLNDPM	56(15)	174(41)	164(44)	179(42)	748(190)

Tableau 8.13: Nombre moyen de pas pour les différents AIRL non déterministes. Les valeurs entre parenthèses correspondent à l'écart-type.

8.6.3 Nombre de solutions mémorisées

D'un point de vue pratique, la comparaison des algorithmes conduit à l'évaluation des ressources nécessaires à leur fonctionnement. Dans la section précédente, nous avons mesuré le nombre d'appels

à la fonction coût plutôt que la durée d'exécution. Pour des raisons similaires[10], nous mesurons le nombre de solutions mémorisées plutôt que l'occupation mémoire effective. Pour la suite des calculs, nous considérons des solutions en codage direct : du fait de leur coût en terme d'occupation mémoire, elles correspondent à la quasi-totalité de la mémoire occupée par l'algorithme considéré.

En ce qui concerne les AIRL, il suffit en général de mémoriser trois solutions : la solution courante s, le voisin courant s' et la meilleure solution trouvée jusqu'alors s^*. Il n'est pas nécessaire de mémoriser davantage de solutions pour les AIRL dotés de mémoire de mouvements (AIRLNDM, tabou, ...) car bien souvent un nombre entier suffit à identifier un mouvement à partir d'une numérotation des mouvements prédéfinie. Par conséquent, pour les algorithmes utilisés dans notre étude, l'occupation en mémoire de la liste des mouvements mémorisés est négligeable par rapport à la taille occupée en mémoire par une solution.

En ce qui concerne les algorithmes génétiques, il faut $2+2p$ solutions pour un AG générationnel et $2+p$ pour un AG stationnaire : deux solutions sont utilisées temporairement pour appliquer le schéma d'application à deux individus, il y a ensuite une ou deux populations (selon le type d'AG) constituées de p individus.

8.6.4 Amélioration des résultats

Bien que les solutions fournies par l'algorithme génétique hybride AG[]+Tabou soient de très bonne qualité (voir section 8.5.1.1, page 151), nous sommes en mesure de fournir un certain nombre de pistes visant à améliorer les performances obtenues à la fois en terme de qualité de solution finale et de fréquence de découverte de solutions de bonne qualité.

Nos travaux, menés jusqu'alors sur deux fronts pourraient se compléter au sein de cet hybride. En effet, nous avons étudié l'hybridation de méthodes de recherche pour aboutir aux hybrides présentés. Dans le même temps, nous avons étudié le paysage du *jobshop* pour aboutir à la définition de fonctions multicritères.

Lors d'une première tentative d'utilisation des fonctions multicritères au sein de l'hybride, nous nous sommes heurtés aux problèmes liés aux déplacements sur les plateaux, aux interactions entre mutation et crossover, et enfin aux problèmes liés au fonctionnement de l'opérateur GA/GT incompatible avec les fonctions multicritères.

Nous avons repris et étudié séparément chacun de ces problèmes, les résultats de cette étude font l'objet de ce mémoire. Nous sommes désormais en mesure d'assembler ces techniques afin d'améliorer les résultats de l'hybride.

Le paramétrage et les tests de cet hybride nécessiteraient une étude complémentaire désormais située hors du cadre de nos recherches actuelles.

8.7 Conclusion

L'étude du paysage du *jobshop* nous a amené à définir des fonctions objectifs multicritères. Parallèlement à cette étude nous avons conçu plusieurs algorithmes hybrides en nous basant sur les caractéristiques intrinsèques des méthodes de recherche utilisées. Nous avons également défini et utilisé un schéma d'application des opérateurs non traditionnel, suite à la mise en évidence des interactions entre les opérateurs d'un algorithme évolutif. Finalement nous avons abordé l'étude de l'intégration de l'ensemble des techniques et présenté des résultats relatifs à un algorithme évolutif non hybride, utilisant à la fois un schéma d'application non standard et des fonctions multicritères.

[10] Biais introduits par les calculateurs/processeurs, les compilateurs ou les systèmes d'exploitation utilisés, ou encore par le type de code (canonique ou optimisé), etc.

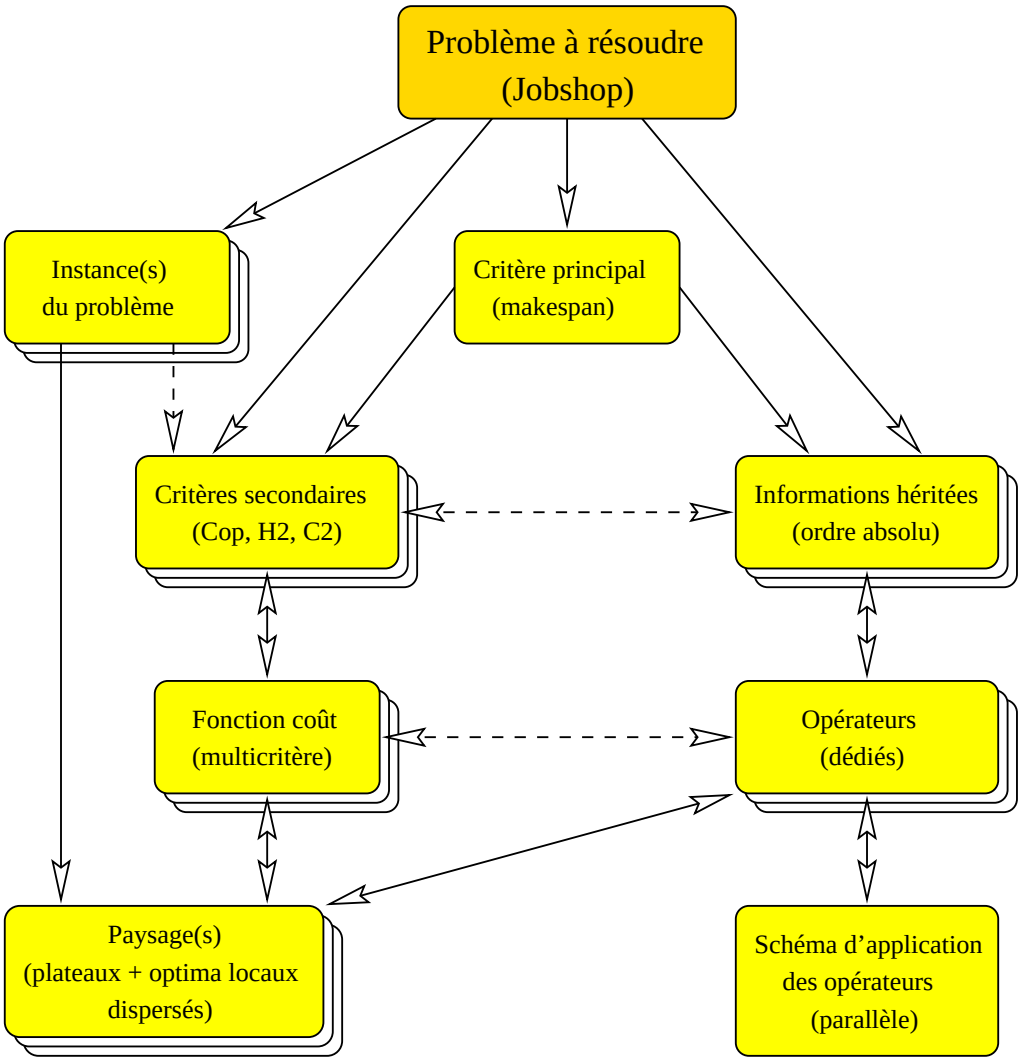


Figure 8.10: Schéma général d'application d'une méthode de recherche à un problème : le problème à résoudre implique la définition du critère principal en terme de coût à optimiser. Ce critère est bien souvent utilisé seul en guise de fonction coût. Les propriétés – ou particularités – de certaines instances à résoudre ou du critère principal peuvent conduire à la définition de critères secondaires (pour le *jobshop* simple, le manque de *discernement* des solutions en terme de makespan). La première étape vers la spécialisation de ces méthodes consiste à modifier les opérateurs utilisés afin de transmettre aux solutions successivement engendrées les informations spécifiques au problème traité (ordre relatif, ordre absolu, adjacence, position, ...). Parallèlement à cette modification, la spécialisation d'une méthode de recherche peut passer par l'intégration de critères secondaires dans la fonction coût et dans les opérateurs (par exemple, les opérations critiques (C_{op}) pour le problème de *jobshop* sont utilisables à la fois dans les opérateurs et dans la fonction coût). Une fois la fonction coût et les opérateurs définis, l'étude du paysage adaptatif associé aux instances à optimiser peut donner des informations sur le type de méthode de recherche susceptible de fournir les meilleurs résultats (en terme de qualité de solution finale ou de rapidité d'exécution). Le choix des opérateurs, peut également influencer le choix du schéma d'application des opérateurs (lorsque la méthode de recherche utilisée comporte plusieurs opérateurs).

Ces résultats montrent qu'une étude complémentaire serait nécessaire pour caractériser plus précisément le rôle de chaque composant dans ces interactions.

À l'exception des méthodes hybrides, le schéma 8.10 synthétise l'ensemble de nos travaux. Il montre que l'intégration d'informations dédiées a lieu à tous les niveaux, si bien que partant de méthodes de recherche générales, nous aboutissons à des méthodes dédiées.

Comme indiqué sur le schéma 8.10, les choix effectués ne sont pas « à sens unique » d'un niveau vers le suivant. Au contraire, les choix effectués à un niveau peuvent remettre en cause les choix effectués précédemment (à un niveau antérieur) : par exemple, si le paysage obtenu présente des caractéristiques non souhaitables, le choix des critères intégrés dans la fonction coût ou le choix des opérateurs peut être remis en cause (cette remise en cause est symbolisée par des arcs ascendants).

Afin de déterminer les performances de la méthode de recherche obtenue, il est nécessaire de disposer d'instances utilisées par différents auteurs de manière à comparer le plus grand nombre d'algorithmes entre-eux. Nous avons utilisé pour cela des instances de la *OR-Library*.

Chapitre 9

Conclusion générale

À l'issue d'une comparaison de différents types de codages et d'opérateurs, nous avons défini un codage et des opérateurs adaptés au *jobshop*. Une étude des composants des méthodes de recherche utilisées nous a permis d'améliorer leurs performances. Nous avons montré expérimentalement qu'une fonction coût uniquement basée sur le makespan est insuffisante pour guider efficacement l'algorithme dans l'espace de recherche. Aussi nous avons défini une fonction coût, basée principalement sur le makespan, intégrant des informations complémentaires sous forme de critères secondaires. Nous avons mis en œuvre une technique basée sur une mesure de « pouvoir discriminant » et de corrélation pour déterminer l'utilité des critères utilisés. Nous avons également étudié l'impact du schéma d'application des opérateurs d'un algorithme évolutif sur les performances obtenues en fonction des opérateurs utilisés.

L'étude menée à montré que l'utilisation d'une fonction « multicritère » n'est pas toujours souhaitable lorsque la méthode de recherche utilisée est capable de se déplacer sur les plateaux. C'est le cas notamment des algorithmes évolutifs, pour lesquels bien que les fonctions coût multicritères augmentent la diversité de la population de manière artificielle (ce qui normalement se traduit par une amélioration des performances), l'ajout des critères secondaires dégrade les résultats obtenus pour certains paramétrages de l'opérateur de recombinaison GA/GT.

Bien qu'initialement destinée à la réduction du nombre d'appels à la fonction coût, l'utilisation de voisinages restreints permet d'atténuer ces effets indésirables en réduisant la taille des plateaux.

La notion de paysage adaptatif s'est avérée très utile à ce niveau, plutôt en tant que représentation de l'espace de recherche (notion de plateaux, de vallées, de massif central) que comme outil d'investigation (longueur de corrélation, corrélation coût-distance, ...).

La comparaison des résultats de plusieurs heuristiques (AE, AIRL et méthode tabou) met en évidence différents « niveaux de difficulté » pour les instances du *jobshop*. Aussi nous avons défini plusieurs critères dans le but de créer un indicateur, qui nous renseigne *a priori* sur la difficulté d'une instance du *jobshop*. Les résultats actuels permettent de déterminer si une instance a tendance à être facilement résolue. En revanche, nous ne sommes pas encore parvenu à coupler une mesure de difficulté avec l'étude du paysage de l'espace de recherche pour sélectionner, avec une certaine précision, l'heuristique (ou le paramétrage) à utiliser pour résoudre efficacement une instance donnée.

Nous avons conçu plusieurs algorithmes évolutifs hybrides grâce auxquels nous obtenons de meilleurs résultats par rapport à l'AE non-hybride. Les résultats coïncident avec ceux de nombreux auteurs qui ont constaté l'amélioration des performances lors de l'hybridation des algorithmes évolutifs avec d'autres méta-heuristiques ou des heuristiques dédiées au *jobshop*. Notre algorithme évolutif hybride engendre des solutions optimales ou quasi-optimales sur une large gamme d'instances de tailles très variées.

Les techniques utilisées pour le *jobshop* sont parfaitement applicables à d'autres problèmes : certains points sont indépendants du problème traité (hybridation, schéma d'application des opérateurs, évolution sur les plateaux, voisinage non constant), pour les autres points (choix du codage, critères secondaires, étude du paysage) la démarche suivie pour le *jobshop* peut être généralisée à d'autres problèmes.

L'étude menée a montré l'existence de nombreuses interactions entre les composants d'une technique de recherche. Nous avons étudié quelques méthodes hybrides « séquentielles ». L'étude de la coévolution de plusieurs méthodes de recherche fait partie de nos perspectives ...

9.1 Perspectives

De nombreuses pistes ont été explorées, notamment au début de nos travaux, certaines méritent d'être poursuivies. Nous en présentons quelques unes dans cette section.

9.1.1 Colonies de Fourmis et *jobshop*

Les colonies de Fourmis ont déjà montré leur efficacité dans divers domaines. Dans le cadre du groupe PERFORM, nous avons évalué cette méthode pour le QAP et le TSP [Rou98, TRFR99, RFRT98] : les résultats obtenus sont très prometteurs. Nous envisageons d'appliquer cette technique au *jobshop*.

9.1.2 Fonctions coût multicritères

Nous avons surtout insisté sur les critères (H_2 et C_{op}) pour illustrer notre démarche. Cependant, il existe bien d'autres critères utilisables dans le cadre de la résolution du *jobshop*. Il serait intéressant de mener une étude complémentaire sur ces critères afin d'évaluer dans quelle mesure ils sont capables d'améliorer les résultats.

C'est le cas, par exemple, du critère C_2 que nous avons défini dans le cadre de nos travaux et utilisé dans la fonction f_6. Ce critère soulève un autre point intéressant : sur certaines instances il permet à f_6 de surpasser les autres fonctions, alors que sur d'autres instances il offre de moins bonnes performances. Il semble alors légitime de s'interroger sur l'existence d'indicateurs (comme les ratio $\frac{J}{M}$ ou $\frac{P_{\max}}{\Pi_{\max}}$) capables de sélectionner (par exemple entre H_2 et C_2) les critères à utiliser.

9.1.3 Borne supérieure pour le makespan

Pour définir nos critères relatifs à la difficulté d'une instance, nous utilisons actuellement des bornes qui nous fournissent un encadrement de l'intervalle des valeurs de makespan relatif à l'ensemble des ordonnancements sans-délai. Nous envisageons d'intégrer des bornes plus précises afin de fournir un encadrement de la valeur optimale du makespan. Ces bornes sont en effet très importantes pour les méthodes de recherche de type *branch and bound* : les instances pour lesquelles des bornes (statiques ou dynamiques) proches de l'optimum peuvent être rapidement obtenues, sont facilement résolues par ces méthodes en raison de l'élagage de l'arbre de recherche induit par ces bornes. Par conséquent, elles fournissent (indirectement) des indications sur la difficulté d'une instance. Nous envisageons de les intégrer ultérieurement dans le calcul de nos critères.

Si nous nous référons aux résultats concernant les algorithmes approchés pour le *jobshop*, nous pouvons obtenir des valeurs de makespan situées à moins de $\rho\times$ valeur_optimale, en garantissant ce résultats pour toutes les instances du *jobshop*. Il suffit pour cela d'intégrer un algorithme approché dans la phase d'initialisation de l'AG (hybride). Les résultats théoriques précisent toutefois que la

valeur de ρ ne peut pas être inférieure à $\frac{5}{4}$ (sauf si $\mathcal{P} = \mathcal{NP}$). Un tel algorithme permet de définir une borne supérieure de l'écart entre la solution trouvée et la solution optimale, et de calculer les performances minimales de l'algorithme en terme de qualité de solution finale.

Nous travaillons actuellement sur l'utilisation simultanée de plusieurs heuristiques dans l'algorithme génétique hybride, notamment au sein de la phase d'initialisation. L'objectif consiste à accélérer l'algorithme tout en améliorant la qualité des solutions obtenues.

9.1.4 Amélioration des résultats

Partant de plusieurs méta-heuristiques dans leur version « canonique » dans le (seul) but de comprendre leur fonctionnement et d'expliquer leurs évolutions dans un paysage comportant de nombreux plateaux et optima locaux, nous sommes parvenus à obtenir par hybridation un algorithme capable d'engendrer des solutions optimales ou quasi-optimales.

L'objectif initial n'étant pas la recherche de la performance, il est clair que les algorithmes mis en œuvre peuvent être améliorés en terme de qualité de solution finale, mais surtout en terme de durée d'exécution. C'est particulièrement vrai pour la recherche tabou utilisée dans le cadre de notre algorithme hybride : nous sommes parfaitement conscient du fait qu'il existe des versions extrêmement performantes de cette méta-heuristique qu'il serait inconcevable de négliger dans un nouvel algorithme hybride performant.

En ce qui concerne l'algorithme génétique utilisé pour réaliser notre hybride, il s'agit davantage d'améliorer le code que de revoir le principe de l'algorithme en lui même. En revanche, bien que la bibliothèque d'algorithmes génétiques *LibGA* offre la possibilité d'utiliser un algorithme génétique stationnaire, nous nous sommes limités à un algorithme génétique de type générationnel dans le cadre de nos travaux. Il serait donc intéressant d'évaluer l'apport de l'algorithme génétique stationnaire au sein d'une méthode hybride.

Une autre amélioration de l'algorithme génétique utilisé concerne l'opérateur GA/GT : dans sa version actuelle cet opérateur optimise le makespan sans tenir compte des critères secondaires. La définition de nouveaux opérateurs intégrant ces critères devrait améliorer les performances obtenues.

Partie IV

Annexes

Annexe A

Résultats complémentaires

Cette annexe rassemble divers résultats détaillés non présentés dans les chapitres de ce mémoire afin de ne pas entrecouper le texte par de longues tables de résultats.

A.1 Résultats détaillés de l'AIRLNDM

Les tableaux A.1, A.2, A.3 et A.4 présentent les résultats obtenus respectivement par l'AIRLNDM(f_1), l'AIRLNDM(f_2), l'AIRLNDM(f_3) et l'AIRLNDM(f_6) sur les instances de la *OR-Library*. La colonne intitulée $M\log(J!)$ donne une estimation du logarithme de la taille de l'espace de recherche (voir 7.2.1, page 91). La colonne intitulée $\%_{\mathsf{opt}}$ contient l'écart (en %) de la moyenne du makespan des solutions finales à la valeur du makespan optimum (ou à sa borne inférieure si ce dernier n'est pas connu) ; cette mesure est complétée par les colonnes (min, moy, max, écart, exp) qui donnent respectivement la valeur du makespan minimum, de la moyenne du makespan, du makespan maximum, de l'écart-type, et du nombre d'expériences lancées. Enfin la colonne pas correspond au nombre de pas moyen effectués avant arrêt de l'AIRLNDM sur un optimum local (ou un plateau). Les instances sont classées par ordre croissant en fonction de la colonne $\%_{\mathsf{opt}}$. Une ligne horizontale de séparation est positionnée à $5\%_{\mathsf{opt}}$, $10\%_{\mathsf{opt}}$, $15\%_{\mathsf{opt}}$ et $20\%_{\mathsf{opt}}$.

$\%_{\mathsf{opt}}$	$M\log(J!)$	Inst	JxM	Optimum	min	moy	max	écart	exp	pas
15.79	212	ft20	20x5	1165	1239	1349.00	1509	43	500	25.16
16.56	151	ft10	10x10	930	991	1084.00	1233	42	500	15.64
18.60	418	ta01	15x15	1231	1351	1460.00	1600	47	500	15.34
19.05	635	abz7	20x15	656	738	781.10	853	19	500	17.98
21.88	1120	ta31	30x15	1764	2020	2150.00	2339	47	500	24.72

Tableau A.1: Résultats de l'AIRLNDM(f_1).

$\%_{\mathsf{opt}}$	$M\log(J!)$	Inst	JxM	Optimum	min	moy	max	écart	exp	pas
0.00	139	la06	15x5	926	926	926.00	926	0.0000	100	24.44
0.00	212	la14	20x5	1292	1292	1292.00	1292	0.0000	100	32.42
0.00	212	la11	20x5	1222	1222	1222.00	1250	2.7960	100	32.76
0.02	76	la05	10x5	593	593	593.10	596	0.4200	100	16.81
0.02	139	la10	15x5	958	958	958.20	971	1.3094	100	26.14
0.09	212	la13	20x5	1150	1150	1151.00	1184	4.6577	100	39.30

Résultats de l'AIRLNDM(f_2) .../...

%opt	$M\log(J!)$	Inst	JxM	Optimum	min	moy	max	écart	exp	pas
0.10	212	la12	20x5	1039	1039	1040.00	1066	3.3012	100	41.07
0.14	139	la09	15x5	951	951	952.30	971	3.8643	100	32.06
1.27	139	la08	15x5	863	863	874.00	927	13.7517	100	32.56
2.03	139	la07	15x5	890	890	908.10	1046	25.9909	100	24.12
2.73	747	la33	30x10	1719	1719	1766.00	1864	35.1424	100	96.15
2.91	747	la31	30x10	1784	1784	1836.00	1929	31.0326	100	105.20
3.07	212	la15	20x5	1207	1207	1244.00	1344	31.4079	100	55.20
3.69	76	la01	10x5	666	666	690.60	792	24.1692	100	15.17
4.00	747	la32	30x10	1850	1850	1924.00	2017	32.9403	100	98.91
5.56	747	la35	30x10	1888	1908	1993.00	2187	49.8006	100	86.53
6.33	747	la34	30x10	1721	1753	1830.00	1940	40.9641	100	111.20
7.08	76	la04	10x5	590	593	631.80	722	26.5249	100	19.11
7.53	151	abz6	10x10	943	954	1014.00	1128	31.6100	100	24.33
7.54	151	la17	10x10	784	787	843.10	995	36.6707	100	24.37
7.94	151	la16	10x10	945	979	1020.00	1100	27.7904	100	19.64
8.33	279	la23	15x10	1032	1052	1118.00	1237	40.5429	100	39.61
8.49	151	la20	10x10	902	920	978.60	1055	31.9869	100	21.83
8.79	76	la03	10x5	597	606	649.50	725	23.1661	100	18.75
8.91	151	abz5	10x10	1234	1262	1344.00	1587	47.4475	100	19.98
9.30	423	la30	20x10	1355	1407	1481.00	1600	37.8891	100	74.27
9.48	151	la18	10x10	848	859	928.40	1038	36.0965	100	22.04
9.76	76	la02	10x5	655	655	718.90	799	31.1578	100	22.55
10.68	151	la19	10x10	842	875	931.90	1023	30.4305	100	22.95
10.85	7275	ta71	100x20	5464	5938	6057.00	6167	59.3114	21	491.10
11.22	91	car5	10x6	7702	7868	8566.00	10056	438.9180	100	16.11
11.35	151	orb9	10x10	934	978	1040.00	1138	31.9869	100	27.01
11.85	60	car7	7x7	6558	6558	7335.00	8339	426.1940	100	14.45
12.19	85	car8	8x8	8264	8477	9271.00	10453	392.7710	100	15.99
12.32	423	la26	20x10	1218	1273	1368.00	1469	42.8278	100	65.54
12.33	151	orb2	10x10	888	923	997.50	1158	37.5306	100	25.23
12.43	101	car4	14x4	8003	8129	8998.00	10349	475.7120	100	24.98
12.44	151	orb4	10x10	1005	1061	1130.00	1228	31.1070	100	27.33
13.60	151	orb7	10x10	397	410	451.00	500	16.8030	100	23.37
13.64	418	la36	15x15	1268	1356	1441.00	1547	38.2098	100	44.67
14.02	279	la25	15x10	977	1048	1114.00	1243	36.0696	100	39.28
14.16	212	ft20	20x5	1165	1211	1330.00	1473	47.8593	500	61.46
14.44	279	la21	15x10	1046	1128	1197.00	1279	35.6357	100	38.70
14.80	423	la28	20x10	1216	1286	1396.00	1530	41.5457	100	67.21
14.82	423	la27	20x10	1235	1348	1418.00	1525	39.0153	100	59.61
14.84	151	ft10	10x10	930	955	1068.00	1180	38.5082	500	30.47
15.03	88	car1	11x5	7038	7091	8096.00	9264	516.8130	100	26.66
15.11	151	orb5	10x10	887	927	1021.00	1158	45.5095	100	21.14
15.19	279	la24	15x10	935	983	1077.00	1169	35.3100	100	43.97
15.24	90	car2	13x4	7166	7427	8258.00	9353	420.5740	100	25.32
15.46	418	la37	15x15	1397	1521	1613.00	1722	46.4201	100	43.58
15.68	418	ta02	15x15	1244	1361	1439.00	1556	43.5124	100	45.57
16.03	418	ta07	15x15	1223/1228	1324	1419.00	1629	46.5448	100	39.01
16.34	100	car3	12x5	7312	7543	8507.00	9899	536.7010	100	23.04

Résultats de l'AIRLNDM(f_2) .../...

$\%_{opt}$	$M\log(J!)$	Inst	JxM	Optimum	min	moy	max	écart	exp	pas
16.71	151	orb1	10x10	1059	1142	1236.00	1324	41.3785	100	30.80
16.79	418	la39	15x15	1233	1346	1440.00	1610	46.3372	100	43.34
16.84	418	ta10	15x15	1241	1362	1450.00	1568	45.2096	100	42.85
16.94	279	la22	15x10	927	965	1084.00	1202	47.1359	100	48.46
17.03	635	ta14	20x15	1345	1488	1574.00	1682	42.2105	100	56.42
17.13	151	orb8	10x10	899	955	1053.00	1184	42.9307	100	28.82
17.14	418	ta01	15x15	1231	1334	1442.00	1606	45.0897	500	52.65
17.27	418	la40	15x15	1222	1339	1433.00	1579	46.3177	100	42.87
17.33	102	car6	9x8	8313	8637	9754.00	11260	453.8970	100	15.58
17.33	151	orb6	10x10	1010	1077	1185.00	1338	51.4252	100	31.35
17.69	418	ta06	15x15	1210/1239	1331	1424.00	1569	42.9914	100	39.94
17.94	635	abz7	20x15	656	732	773.70	836	18.5550	500	52.31
18.22	151	orb10	10x10	944	1005	1116.00	1306	53.2074	100	22.06
18.48	423	la29	20x10	1142/1153	1283	1353.00	1428	31.6722	100	60.47
18.74	418	ta03	15x15	1206/1218	1340	1432.00	1528	43.9548	100	42.21
18.89	418	ta04	15x15	1170/1175	1298	1391.00	1531	41.7269	100	41.96
19.26	418	ta05	15x15	1210/1228	1337	1443.00	1582	45.0323	100	44.16
19.28	2970	ta61	50x20	2868	3337	3421.00	3586	44.6430	100	185.60
19.34	635	ta17	20x15	1458/1464	1601	1740.00	1873	51.8761	100	77.30
19.48	418	la38	15x15	1196	1301	1429.00	1579	44.2669	100	41.71
20.29	2227	ta51	50x15	2760	3180	3320.00	3491	63.2742	100	277.70
20.69	1120	ta31	30x15	1764/1766	2020	2129.00	2271	44.2450	500	128.60
20.97	847	ta24	20x20	1602/1651	1835	1938.00	2074	44.0323	100	67.59
21.00	151	orb3	10x10	1005	1087	1216.00	1337	47.2744	100	29.21
21.01	418	ta09	15x15	1247/1274	1405	1509.00	1621	45.6033	100	44.13
21.15	418	ta08	15x15	1187/1217	1321	1438.00	1565	45.0303	100	42.73
22.94	847	ta28	20x20	1591/1615	1818	1956.00	2072	48.5289	100	66.26
23.08	635	ta18	20x15	1369/1396	1599	1685.00	1785	42.7249	100	71.98
23.78	635	ta20	20x15	1316/1353	1530	1629.00	1754	43.1829	100	64.33
23.94	635	abz8	20x15	645/669	747	799.40	856	20.4899	100	56.80
25.22	635	abz9	20x15	661/679	777	827.70	899	22.2085	100	71.25
25.74	635	ta11	20x15	1321/1364	1567	1661.00	1772	43.7824	100	69.82
26.15	635	ta16	20x15	1300/1368	1538	1640.00	1777	41.1403	100	56.02
26.19	635	ta12	20x15	1321/1367	1575	1667.00	1851	52.6701	100	63.84
26.96	635	ta19	20x15	1276/1341	1526	1620.00	1757	38.7685	100	68.75
27.13	847	ta25	20x20	1504/1598	1830	1912.00	2066	53.0770	100	64.23
27.14	635	ta13	20x15	1271/1350	1525	1616.00	1771	43.7358	100	58.62
27.53	635	ta15	20x15	1293/1342	1571	1649.00	1784	42.2105	100	68.55
27.97	847	ta27	20x20	1616/1689	1902	2068.00	2213	55.9204	100	67.87
28.26	847	ta23	20x20	1472/1558	1796	1888.00	2023	47.4724	100	66.35
28.92	847	ta22	20x20	1511/1601	1847	1948.00	2179	52.4869	100	68.41
29.43	847	ta26	20x20	1539/1655	1900	1992.00	2111	44.0681	100	57.73
30.12	847	ta29	20x20	1514/1625	1823	1970.00	2159	58.2631	100	72.30
32.38	847	ta30	20x20	1473/1596	1836	1950.00	2076	51.4648	100	59.67
36.47	1493	ta41	30x20	1859/2023	2445	2537.00	2667	45.6933	100	133.20
45.70	847	ta21	20x20	1359/1647	1863	1980.00	2086	44.7676	100	68.96

Tableau A.2: Résultats de l'AIRLNDM(f_2)

$\%_{opt}$	$M\log(J!)$	Inst	JxM	Optimum	min	moy	max	écart	exp	pas
0.00	139	la06	15x5	926	926	926.00	926	0.0000	100	24.93
0.00	212	la11	20x5	1222	1222	1222.00	1224	0.2800	100	32.66
0.00	212	la14	20x5	1292	1292	1292.00	1292	0.0000	100	33.38
0.00	212	la12	20x5	1039	1039	1039.00	1058	1.9151	100	43.55
0.01	139	la10	15x5	958	958	958.10	971	1.3067	100	25.85
0.02	76	la05	10x5	593	593	593.10	596	0.4200	100	16.88
0.06	139	la09	15x5	951	951	951.60	971	2.3846	100	32.86
0.09	212	la13	20x5	1150	1150	1151.00	1171	2.5278	100	42.25
1.02	139	la08	15x5	863	863	871.80	927	12.6770	100	33.92
1.56	139	la07	15x5	890	890	903.90	1046	22.9808	100	26.23
2.50	747	la33	30x10	1719	1719	1762.00	1864	33.5269	100	105.50
2.86	747	la31	30x10	1784	1786	1835.00	1941	30.9640	100	112.40
3.15	212	la15	20x5	1207	1207	1245.00	1344	32.0552	100	53.91
3.45	76	la01	10x5	666	666	689.00	792	23.3957	100	15.99
3.78	747	la32	30x10	1850	1850	1920.00	2014	33.5040	100	105.00
5.22	39	ft6	6x6	55	55	57.87	70	2.5560	100	9.78
5.67	747	la35	30x10	1888	1908	1995.00	2194	50.0640	100	86.57
5.98	747	la34	30x10	1721	1753	1824.00	1940	39.7782	100	120.60
6.88	76	la04	10x5	590	593	630.60	722	26.8919	100	19.86
7.59	151	la17	10x10	784	786	843.50	995	37.8884	100	24.72
7.64	151	abz6	10x10	943	954	1015.00	1128	32.4770	100	24.25
7.83	151	la16	10x10	945	975	1019.00	1100	27.5698	100	19.93
8.24	279	la23	15x10	1032	1047	1117.00	1237	40.8089	100	41.43
8.45	151	la20	10x10	902	921	978.20	1055	31.8935	100	21.83
8.67	151	abz5	10x10	1234	1262	1341.00	1587	48.2372	100	20.35
8.98	76	la03	10x5	597	606	650.60	725	22.9120	100	19.03
9.23	151	la18	10x10	848	859	926.30	1038	36.0050	100	22.35
9.40	76	la02	10x5	655	660	716.60	799	31.2096	100	23.17
9.59	423	la30	20x10	1355	1398	1485.00	1600	40.2544	100	72.91
9.82	1120	ta35	30x15	2007	2119	2204.00	2291	37.4948	100	87.93
10.52	7275	ta71	100x20	5464	5914	6039.00	6179	57.1475	100	583.50
10.63	151	la19	10x10	842	877	931.50	1023	29.5566	100	22.89
11.24	151	orb9	10x10	934	970	1039.00	1138	33.3762	100	27.55
11.27	91	car5	10x6	7702	7868	8570.00	10056	445.2850	100	16.26
11.77	60	car7	7x7	6558	6558	7330.00	8339	434.1440	100	14.55
12.08	101	car4	14x4	8003	8129	8970.00	10349	434.5990	100	25.53
12.11	151	orb2	10x10	888	923	995.50	1158	37.5706	100	25.98
12.14	85	car8	8x8	8264	8477	9267.00	10453	403.8100	100	16.19
12.24	151	orb4	10x10	1005	1061	1128.00	1203	29.7463	100	28.41
12.32	423	la26	20x10	1218	1285	1368.00	1469	40.1290	100	67.84
13.53	151	orb7	10x10	397	410	450.70	487	16.2774	100	23.89
13.56	418	la36	15x15	1268	1356	1440.00	1547	37.5358	100	45.63
13.73	212	ft20	20x5	1165	1215	1325.00	1470	45.6026	500	66.25
13.98	423	la28	20x10	1216	1298	1386.00	1530	40.3102	100	70.43
14.02	279	la25	15x10	977	1045	1114.00	1243	35.9411	100	40.49
14.05	279	la21	15x10	1046	1115	1193.00	1279	35.5642	100	42.67

Résultats de l'AIRLNDM(f_3) .../...

$\%_{opt}$	$M\log(J!)$	Inst	JxM	Optimum	min	moy	max	écart	exp	pas
14.73	151	ft10	10x10	930	951	1067.00	1180	38.2091	500	31.12
14.86	418	ta06	15x15	1238	1331	1422.00	1569	41.0634	100	41.69
14.90	423	la27	20x10	1235	1339	1419.00	1525	37.1567	100	61.50
14.97	279	la24	15x10	935	983	1075.00	1169	35.6944	100	44.42
15.06	88	car1	11x5	7038	7038	8098.00	9264	519.8170	100	26.70
15.14	90	car2	13x4	7166	7427	8251.00	9260	415.5060	100	25.16
15.32	418	ta07	15x15	1227	1322	1415.00	1629	44.6551	100	39.49
15.45	151	orb5	10x10	887	927	1024.00	1158	46.5257	100	21.20
15.53	418	la37	15x15	1397	1503	1614.00	1722	48.0494	100	44.92
15.59	418	ta02	15x15	1244	1361	1438.00	1555	43.1114	100	48.62
16.32	100	car3	12x5	7312	7543	8505.00	9899	528.4810	100	23.18
16.50	279	la22	15x10	927	994	1080.00	1202	43.4213	100	49.80
16.53	151	orb1	10x10	1059	1142	1234.00	1324	41.9221	100	31.18
16.60	418	ta10	15x15	1241	1362	1447.00	1568	46.0886	100	45.65
16.71	418	la39	15x15	1233	1351	1439.00	1610	46.1600	100	43.48
16.91	151	orb8	10x10	899	955	1051.00	1184	40.7740	100	29.56
16.98	418	ta01	15x15	1231	1334	1440.00	1606	43.8476	500	54.58
17.10	635	ta14	20x15	1345	1488	1575.00	1682	42.2271	100	54.98
17.16	418	ta03	15x15	1218	1332	1427.00	1528	46.0016	100	44.84
17.18	418	la40	15x15	1222	1344	1432.00	1579	48.2254	100	44.32
17.33	151	orb6	10x10	1010	1077	1185.00	1338	49.5909	100	31.53
17.51	102	car6	9x8	8313	8637	9769.00	11260	459.1250	100	15.50
17.57	418	ta05	15x15	1224	1337	1439.00	1582	47.0358	100	46.33
17.75	418	ta08	15x15	1217	1339	1433.00	1565	42.0231	100	44.13
17.91	635	abz7	20x15	656	732	773.50	836	18.5220	500	54.35
18.04	423	la29	20x10	1142/1153	1283	1348.00	1427	29.2572	100	63.17
18.29	418	ta09	15x15	1274	1428	1507.00	1646	42.1135	100	45.56
18.30	418	ta04	15x15	1175	1307	1390.00	1531	40.8769	100	42.02
18.33	151	orb10	10x10	944	994	1117.00	1306	54.2062	100	22.61
18.66	1120	ta39	30x15	1795	2037	2130.00	2312	46.4173	100	99.08
19.25	2970	ta61	50x20	2868	3322	3420.00	3586	48.1206	100	187.30
19.41	635	ta17	20x15	1458/1464	1650	1741.00	1873	50.4472	100	79.23
19.48	418	la38	15x15	1196	1322	1429.00	1579	44.7619	100	43.32
19.71	2227	ta51	50x15	2760	3089	3304.00	3431	67.6768	100	321.60
20.02	1120	ta34	30x15	1828/1832	2095	2194.00	2305	43.3614	100	103.80
20.52	1120	ta31	30x15	1764	2020	2126.00	2267	43.4064	500	133.50
20.79	847	ta24	20x20	1602/1651	1850	1935.00	2074	46.8633	100	71.15
21.19	151	orb3	10x10	1005	1087	1218.00	1337	48.9668	100	29.05
22.38	847	ta28	20x20	1591/1615	1818	1947.00	2072	55.6178	100	70.12
22.43	635	ta18	20x15	1369/1396	1603	1676.00	1792	40.9722	100	78.19
22.59	1120	ta36	30x15	1819	2099	2230.00	2403	55.0915	100	110.70
23.04	1120	ta37	30x15	1771/1784	2076	2179.00	2285	43.8126	100	108.00
23.49	1120	ta38	30x15	1673/1677	1935	2066.00	2202	50.5029	100	107.70
23.63	635	ta20	20x15	1316/1353	1530	1627.00	1754	42.4222	100	64.19
23.64	1493	ta45	30x20	1997/2005	2356	2469.00	2609	48.3828	100	98.90
23.81	635	abz8	20x15	645/669	746	798.60	856	19.1872	100	59.41
24.41	1120	ta32	30x15	1774/1803	2098	2207.00	2355	47.5667	100	112.70
24.97	1120	ta33	30x15	1778/1796	2121	2222.00	2383	55.4839	100	144.40

Résultats de l'AIRLNDM(f_3) .../...

$\%_{opt}$	$M\log(J!)$	Inst	JxM	Optimum	min	moy	max	écart	exp	pas
24.98	635	abz9	20x15	661/679	780	826.10	899	23.3591	100	72.95
25.44	635	ta11	20x15	1321/1364	1558	1657.00	1768	44.6486	100	73.00
25.58	1493	ta48	30x20	1912/1971	2308	2401.00	2603	51.9994	100	102.20
26.08	635	ta16	20x15	1300/1362	1538	1639.00	1777	45.3255	100	57.45
26.19	635	ta12	20x15	1321/1367	1565	1667.00	1851	48.8316	100	63.86
26.80	635	ta19	20x15	1276/1341	1523	1618.00	1757	38.1242	100	72.02
26.85	1120	ta40	30x15	1631/1686	1985	2069.00	2166	43.0571	100	90.98
26.99	635	ta13	20x15	1271/1350	1514	1614.00	1771	45.6812	100	61.49
27.19	847	ta25	20x20	1504/1597	1823	1913.00	2066	55.3663	100	66.07
27.76	635	ta15	20x15	1293/1342	1563	1652.00	1789	47.2810	100	66.83
27.85	847	ta27	20x20	1616/1687	1902	2066.00	2213	57.3977	100	68.17
28.04	1493	ta49	30x20	1915/1984	2329	2452.00	2588	50.2125	100	100.90
28.19	847	ta23	20x20	1472/1558	1771	1887.00	2023	49.9495	100	66.92
28.30	1493	ta46	30x20	1940/2029	2369	2489.00	2611	51.3619	100	102.00
28.46	847	ta21	20x20	1539/1645	1861	1977.00	2086	47.5289	100	70.68
28.86	847	ta22	20x20	1511/1601	1838	1947.00	2179	53.2436	100	69.59
28.98	847	ta26	20x20	1539/1651	1900	1985.00	2091	44.0883	100	62.54
29.16	1493	ta44	30x20	1927/1998	2381	2489.00	2614	47.4767	100	95.53
30.05	847	ta29	20x20	1514/1625	1823	1969.00	2159	56.3273	100	72.91
31.28	1493	ta42	30x20	1867/1961	2337	2451.00	2615	54.0550	100	101.00
31.90	1493	ta43	30x20	1809/1879	2233	2386.00	2515	53.7386	100	106.20
32.18	847	ta30	20x20	1473/1585	1830	1947.00	2076	47.4892	100	60.49
33.04	1493	ta47	30x20	1789/1913	2261	2380.00	2516	48.4733	100	111.40
36.15	1493	ta41	30x20	1859/2023	2421	2531.00	2631	44.1927	100	139.90
36.69	1493	ta50	30x20	1807/1937	2339	2470.00	2651	50.3438	100	125.60

Tableau A.3: Résultats de l'AIRLNDM(f_3)

$\%_{opt}$	$M\log(J!)$	Inst	JxM	Optimum	min	moy	max	écart	exp	pas
0.00	139	la06	15x5	926	926	926.00	929	0.3571	100	41.72
0.00	212	la14	20x5	1292	1292	1292.00	1292	0.0000	100	67.72
0.01	139	la10	15x5	958	958	958.10	968	0.9950	100	46.67
0.08	76	la05	10x5	593	593	593.50	610	2.3850	100	18.88
0.08	212	la11	20x5	1222	1222	1223.00	1235	2.0808	100	71.98
0.17	212	la13	20x5	1150	1150	1152.00	1208	7.1270	100	82.54
0.37	139	la09	15x5	951	951	954.50	1001	7.3654	100	48.25
0.38	212	la12	20x5	1039	1039	1043.00	1087	8.6960	100	79.81
1.37	139	la07	15x5	890	890	902.20	967	17.3795	100	52.91
2.16	139	la08	15x5	863	863	881.60	944	18.1402	100	51.27
3.06	76	la01	10x5	666	666	686.40	735	16.9243	100	22.87
3.49	747	la33	30x10	1719	1719	1779.00	1851	32.8763	100	155.10
3.64	747	la31	30x10	1784	1792	1849.00	1928	31.4594	100	181.30
3.73	212	la15	20x5	1207	1207	1252.00	1343	29.5374	100	100.20
4.59	747	la32	30x10	1850	1852	1935.00	2032	36.7746	100	152.70
5.51	747	la35	30x10	1888	1898	1992.00	2099	44.8300	100	174.60
7.50	747	la34	30x10	1721	1776	1850.00	1982	37.1409	100	156.90
7.64	151	abz6	10x10	943	968	1015.00	1085	25.3610	100	25.07
8.04	151	la16	10x10	945	971	1021.00	1106	28.1180	100	22.81
8.62	91	car5	10x6	7702	7862	8366.00	9627	342.4130	100	35.08

Résultats de l'AIRLNDM(f_6) .../...

$\%_{opt}$	$M\log(J!)$	Inst	JxM	Optimum	min	moy	max	écart	exp	pas
8.73	76	la04	10x5	590	602	641.50	761	28.0070	100	27.80
8.75	151	abz5	10x10	1234	1264	1342.00	1446	38.5135	100	23.06
8.90	76	la02	10x5	655	660	713.30	804	29.3530	100	33.81
9.01	279	la23	15x10	1032	1049	1125.00	1219	39.6652	100	50.59
9.19	151	la18	10x10	848	861	925.90	1022	31.2124	100	25.67
9.35	151	la17	10x10	784	784	857.30	984	42.0790	100	27.26
9.36	76	la03	10x5	597	603	652.90	706	22.7187	100	29.04
9.40	151	la20	10x10	902	915	986.80	1089	37.3993	100	24.17
9.69	85	car8	8x8	8264	8407	9065.00	9816	315.0260	100	24.61
9.81	101	car4	14x4	8003	8057	8788.00	9785	344.0800	100	47.81
10.85	423	la30	20x10	1355	1377	1502.00	1649	45.9587	100	89.53
11.24	60	car7	7x7	6558	6558	7295.00	8110	405.4500	100	18.18
11.24	151	orb9	10x10	934	969	1039.00	1153	36.5288	100	34.40
11.37	151	la19	10x10	842	876	937.70	1010	29.1095	100	23.78
11.48	7275	ta71	100x20	5464	5939	6091.00	6255	58.3643	100	756.30
12.06	90	car2	13x4	7166	7297	8030.00	9192	400.5460	100	43.66
12.43	88	car1	11x5	7038	7076	7913.00	9051	482.0130	100	36.24
12.48	151	orb2	10x10	888	914	998.80	1085	35.8294	100	28.01
12.94	151	orb4	10x10	1005	1070	1135.00	1287	36.5059	100	28.56
13.03	100	car3	12x5	7312	7594	8265.00	9678	439.5740	100	41.45
13.14	423	la26	20x10	1218	1293	1378.00	1475	39.7770	100	88.73
13.82	279	la25	15x10	977	1033	1112.00	1219	37.1199	100	54.46
14.03	151	orb7	10x10	397	420	452.70	507	15.4156	100	26.60
14.27	418	la36	15x15	1268	1359	1449.00	1633	47.6078	100	47.21
14.53	102	car6	9x8	8313	8639	9521.00	10376	379.6630	100	27.05
14.66	151	orb5	10x10	887	942	1017.00	1121	39.9378	100	29.62
15.05	423	la28	20x10	1216	1313	1399.00	1519	40.1782	100	96.11
15.11	279	la21	15x10	1046	1146	1204.00	1345	37.7515	100	51.09
15.11	418	ta02	15x15	1244	1323	1432.00	1569	44.2842	100	52.81
15.30	423	la27	20x10	1235	1324	1424.00	1542	42.1773	100	85.00
15.82	418	la37	15x15	1397	1496	1618.00	1777	45.1461	100	59.09
15.94	279	la24	15x10	935	1016	1084.00	1197	36.9930	100	54.90
16.27	418	ta07	15x15	1223/1228	1331	1422.00	1534	41.9718	100	44.47
16.36	418	ta10	15x15	1241	1347	1444.00	1586	46.4830	100	54.16
16.50	279	la22	15x10	927	982	1080.00	1210	49.1180	100	64.41
16.71	151	orb1	10x10	1059	1144	1236.00	1337	39.6678	100	36.54
16.95	635	ta14	20x15	1345	1483	1573.00	1683	41.3484	100	78.46
17.02	418	la40	15x15	1222	1327	1430.00	1553	45.9862	100	51.90
17.36	418	la39	15x15	1233	1360	1447.00	1532	39.5278	100	52.61
17.69	151	orb8	10x10	899	984	1058.00	1173	40.3503	100	37.55
17.72	151	orb6	10x10	1010	1070	1189.00	1338	51.7476	100	37.60
17.77	418	ta06	15x15	1210/1239	1338	1425.00	1523	38.3221	100	46.22
17.80	151	orb10	10x10	944	995	1112.00	1255	50.8399	100	33.66
18.99	418	ta03	15x15	1206/1218	1350	1435.00	1542	37.8696	100	54.77
19.06	418	ta04	15x15	1170/1175	1296	1393.00	1485	39.5297	100	49.03
19.26	418	ta05	15x15	1210/1228	1352	1443.00	1564	42.8366	100	51.13
19.26	423	la29	20x10	1142/1153	1279	1362.00	1454	33.5596	100	81.04
19.48	418	la38	15x15	1196	1330	1429.00	1539	42.8056	100	51.67

Résultats de l'AIRLNDM(f_6) .../...

%opt	$M\log(J!)$	Inst	JxM	Optimum	min	moy	max	écart	exp	pas
19.98	2970	ta61	50x20	2868	3325	3441.00	3618	55.2781	100	286.80
20.00	151	orb3	10x10	1005	1103	1206.00	1343	49.2562	100	40.00
20.03	635	ta17	20x15	1458/1464	1631	1750.00	1937	51.2877	100	90.87
20.69	2227	ta51	50x15	2760	3166	3331.00	3477	62.7466	100	411.40
20.77	418	ta09	15x15	1247/1274	1423	1506.00	1613	44.2939	100	50.28
21.15	418	ta08	15x15	1187/1217	1328	1438.00	1578	48.1080	100	48.95
21.79	847	ta24	20x20	1602/1651	1853	1951.00	2097	43.8461	100	73.16
23.51	847	ta28	20x20	1591/1615	1845	1965.00	2074	51.8220	100	70.44
23.67	635	ta18	20x15	1369/1396	1604	1693.00	1835	42.6379	100	90.57
24.20	635	abz8	20x15	645/669	743	801.10	853	20.7206	100	72.34
24.62	635	ta20	20x15	1316/1353	1539	1640.00	1747	47.6885	100	77.76
25.67	635	abz9	20x15	661/679	780	830.70	908	24.4163	100	85.53
25.74	635	ta11	20x15	1321/1364	1555	1661.00	1781	45.2564	100	87.65
26.80	635	ta12	20x15	1321/1367	1555	1675.00	1845	55.2299	100	72.04
26.85	635	ta16	20x15	1300/1368	1539	1649.00	1772	44.5694	100	70.76
27.19	635	ta19	20x15	1276/1341	1504	1623.00	1720	40.9233	100	84.75
27.46	847	ta25	20x20	1504/1598	1793	1917.00	2105	55.5866	100	78.05
28.00	635	ta15	20x15	1293/1342	1527	1655.00	1787	49.0208	100	82.65
28.28	847	ta27	20x20	1616/1689	1921	2073.00	2248	59.8071	100	77.27
28.48	635	ta13	20x15	1271/1350	1545	1633.00	1771	45.4519	100	70.39
29.14	847	ta23	20x20	1472/1558	1797	1901.00	2103	48.6091	100	74.35
29.58	847	ta22	20x20	1511/1601	1866	1958.00	2077	45.6282	100	72.79
30.15	847	ta26	20x20	1539/1655	1911	2003.00	2150	49.4600	100	69.87
30.98	847	ta29	20x20	1514/1625	1882	1983.00	2161	55.3339	100	75.37
32.18	847	ta30	20x20	1473/1596	1850	1947.00	2090	54.4067	100	71.78
37.22	1493	ta41	30x20	1859/2023	2433	2551.00	2657	47.7289	100	158.60
45.84	847	ta21	20x20	1359/1647	1835	1982.00	2114	48.3681	100	75.93

Tableau A.4: Résultats de l'AIRLNDM(f_6)

Le tableau A.5 présente une comparaison entre AIRLNDM(f_2) et AIRLNDM(f_3) sur près de 100 instances. Pour chaque instance, 100 expériences sont effectuées (500 expériences pour ft10, ft20, ta01, abz7, ta31). Trois critères de comparaison sont présentés : il s'agit des performances minimales (meilleure solution trouvée) et moyennes. Le troisième critère concerne l'écart-type associé aux performances moyennes. Pour ces trois critères, le signe $>$ signifie que f_3 donne de meilleurs résultats que f_2, le signe $<$ indique de meilleurs résultats pour f_2 par rapport à f_3, et finalement $=$ signale des performances identiques pour f_2 et f_3.

	AIRLNDM(f_2)			AIRLNDM(f_3)			Comparaison		
Inst	min	moy	écart	min	moy	écart	min	moy	écart
abz5	1262	1344.00	47.4475	1262	1341.00	48.2372	=	>	<
abz6	954	1014.00	31.6100	954	1015.00	32.4770	=	<	<
abz7	732	773.70	18.5550	732	773.50	18.5220	=	>	>
abz8	747	799.40	20.4899	746	798.60	19.1872	>	>	>
abz9	777	827.70	22.2085	780	826.10	23.3591	<	>	<
car1	7091	8096.00	516.8130	7038	8098.00	519.8170	>	<	<
car2	7427	8258.00	420.5740	7427	8251.00	415.5060	=	>	>
car3	7543	8507.00	536.7010	7543	8505.00	528.4810	=	>	>
car4	8129	8998.00	475.7120	8129	8970.00	434.5990	=	>	>
car5	7868	8566.00	438.9180	7868	8570.00	445.2850	=	<	<

Comparaisons entre AIRLNDM(f_2) et AIRLNDM(f_3) .../...

	AIRLNDM(f_2)			AIRLNDM(f_3)			Comparaison		
Inst	min	moy	écart	min	moy	écart	min	moy	écart
car6	8637	9754.00	453.8970	8637	9769.00	459.1250	=	<	<
car7	6558	7335.00	426.1940	6558	7330.00	434.1440	=	>	<
car8	8477	9271.00	392.7710	8477	9267.00	403.8100	=	>	<
ft10	955	1068.00	38.5082	951	1067.00	38.2091	>	>	>
ft20	1211	1330.00	47.8593	1215	1325.00	45.6026	<	>	>
la01	666	690.60	24.1692	666	689.00	23.3957	=	>	>
la02	655	718.90	31.1578	660	716.60	31.2096	<	>	<
la03	606	649.50	23.1661	606	650.60	22.9120	=	<	>
la04	593	631.80	26.5249	593	630.60	26.8919	=	>	<
la05	593	593.10	0.4200	593	593.10	0.4200	=	=	=
la06	926	926.00	0.0000	926	926.00	0.0000	=	=	=
la07	890	908.10	25.9909	890	903.90	22.9808	=	>	>
la08	863	874.00	13.7517	863	871.80	12.6770	=	>	>
la09	951	952.30	3.8643	951	951.60	2.3846	=	>	>
la10	958	958.20	1.3094	958	958.10	1.3067	=	>	>
la11	1222	1222.00	2.7960	1222	1222.00	0.2800	=	=	>
la12	1039	1040.00	3.3012	1039	1039.00	1.9151	=	>	>
la13	1150	1151.00	4.6577	1150	1151.00	2.5278	=	=	>
la14	1292	1292.00	0.0000	1292	1292.00	0.0000	=	=	=
la15	1207	1244.00	31.4079	1207	1245.00	32.0552	=	<	<
la16	979	1020.00	27.7904	975	1019.00	27.5698	>	>	>
la17	787	843.10	36.6707	786	843.50	37.8884	>	<	<
la18	859	928.40	36.0965	859	926.30	36.0050	=	>	>
la19	875	931.90	30.4305	877	931.50	29.5566	<	>	>
la20	920	978.60	31.9869	921	978.20	31.8935	<	>	>
la21	1128	1197.00	35.6357	1115	1193.00	35.5642	>	>	>
la22	965	1084.00	47.1359	994	1080.00	43.4213	<	>	>
la23	1052	1118.00	40.5429	1047	1117.00	40.8089	>	>	<
la24	983	1077.00	35.3100	983	1075.00	35.6944	=	>	<
la25	1048	1114.00	36.0696	1045	1114.00	35.9411	>	=	>
la26	1273	1368.00	42.8278	1285	1368.00	40.1290	<	=	>
la27	1348	1418.00	39.0153	1339	1419.00	37.1567	>	<	>
la28	1286	1396.00	41.5457	1298	1386.00	40.3102	<	>	>
la29	1283	1353.00	31.6722	1283	1348.00	29.2572	=	>	>
la30	1407	1481.00	37.8891	1398	1485.00	40.2544	>	<	<
la31	1784	1836.00	31.0326	1786	1835.00	30.9640	<	>	>
la32	1850	1924.00	32.9403	1850	1920.00	33.5040	=	>	<
la33	1719	1766.00	35.1424	1719	1762.00	33.5269	=	>	>
la34	1753	1830.00	40.9641	1753	1824.00	39.7782	=	>	>
la35	1908	1993.00	49.8006	1908	1995.00	50.0640	=	<	<
la36	1356	1441.00	38.2098	1356	1440.00	37.5358	=	>	>
la37	1521	1613.00	46.4201	1503	1614.00	48.0494	>	<	<
la38	1301	1429.00	44.2669	1322	1429.00	44.7619	<	=	<
la39	1346	1440.00	46.3372	1351	1439.00	46.1600	<	>	>
la40	1339	1433.00	46.3177	1344	1432.00	48.2254	<	>	<
orb1	1142	1236.00	41.3785	1142	1234.00	41.9221	=	>	<
orb10	1005	1116.00	53.2074	994	1117.00	54.2062	>	<	<
orb2	923	997.50	37.5306	923	995.50	37.5706	=	>	<

Comparaisons entre AIRLNDM(f_2) et AIRLNDM(f_3) .../...

	AIRLNDM(f_2)			AIRLNDM(f_3)			Comparaison		
Inst	min	moy	écart	min	moy	écart	min	moy	écart
orb3	1087	1216.00	47.2744	1087	1218.00	48.9668	=	<	<
orb4	1061	1130.00	31.1070	1061	1128.00	29.7463	=	>	>
orb5	927	1021.00	45.5095	927	1024.00	46.5257	=	<	<
orb6	1077	1185.00	51.4252	1077	1185.00	49.5909	=	=	>
orb7	410	451.00	16.8030	410	450.70	16.2774	=	>	>
orb8	955	1053.00	42.9307	955	1051.00	40.7740	=	>	>
orb9	978	1040.00	31.9869	970	1039.00	33.3762	>	>	<
ta01	1334	1442.00	45.0897	1334	1440.00	43.8476	=	>	>
ta02	1361	1439.00	43.5124	1361	1438.00	43.1114	=	>	>
ta03	1340	1432.00	43.9548	1332	1427.00	46.0016	>	>	<
ta04	1298	1391.00	41.7269	1307	1390.00	40.8769	<	>	>
ta05	1337	1443.00	45.0323	1337	1439.00	47.0358	=	>	<
ta06	1331	1424.00	42.9914	1331	1422.00	41.0634	=	>	>
ta07	1324	1419.00	46.5448	1322	1415.00	44.6551	>	>	>
ta08	1321	1438.00	45.0303	1339	1433.00	42.0231	<	>	>
ta09	1405	1509.00	45.6033	1428	1507.00	42.1135	<	>	>
ta10	1362	1450.00	45.2096	1362	1447.00	46.0886	=	>	<
ta11	1567	1661.00	43.7824	1558	1657.00	44.6486	>	>	<
ta12	1575	1667.00	52.6701	1565	1667.00	48.8316	>	=	>
ta13	1525	1616.00	43.7358	1514	1614.00	45.6812	>	>	<
ta14	1488	1574.00	42.2105	1488	1575.00	42.2271	=	<	<
ta15	1571	1649.00	42.2105	1563	1652.00	47.2810	>	<	<
ta16	1538	1640.00	41.1403	1538	1639.00	45.3255	=	>	<
ta17	1601	1740.00	51.8761	1650	1741.00	50.4472	<	<	>
ta18	1599	1685.00	42.7249	1603	1676.00	40.9722	<	>	>
ta19	1526	1620.00	38.7685	1523	1618.00	38.1242	>	>	>
ta20	1530	1629.00	43.1829	1530	1627.00	42.4222	=	>	>
ta21	1863	1980.00	44.7676	1861	1977.00	47.5289	>	>	<
ta22	1847	1948.00	52.4869	1838	1947.00	53.2436	>	>	<
ta23	1796	1888.00	47.4724	1771	1887.00	49.9495	>	>	<
ta24	1835	1938.00	44.0323	1850	1935.00	46.8633	<	>	<
ta25	1830	1912.00	53.0770	1823	1913.00	55.3663	>	<	<
ta26	1900	1992.00	44.0681	1900	1985.00	44.0883	=	>	<
ta27	1902	2068.00	55.9204	1902	2066.00	57.3977	=	>	<
ta28	1818	1956.00	48.5289	1818	1947.00	55.6178	=	>	<
ta29	1823	1970.00	58.2631	1823	1969.00	56.3273	=	>	>
ta30	1836	1950.00	51.4648	1830	1947.00	47.4892	>	>	>
ta31	2020	2129.00	44.2450	2020	2126.00	43.4064	=	>	>
ta41	2445	2537.00	45.6933	2421	2531.00	44.1927	>	>	>
ta51	3180	3320.00	63.2742	3089	3304.00	67.6768	>	>	<
ta61	3337	3421.00	44.6430	3322	3420.00	48.1206	>	>	<
ta71	5938	6057.00	59.3114	5914	6039.00	57.1475	>	>	>

Tableau A.5: Comparaisons entre AIRLNDM(f_2) et AIRLNDM(f_3)

A.2 Résultats détaillés pour les critères définis

Les critères ont été évalués sur les instances de la *OR-Library* grâce à l'outil *Visusch* développé dans le cadre de nos travaux (voir section D.2, page 224). Les résultats obtenus sont rassemblés dans le tableau A.6. Ce tableau, destiné à être complété d'une analyse en composantes principales (ACP), permet de visualiser les valeurs atypiques de certains critères. Pour y parvenir, un code extrêmement simple, basé sur la division du domaine de variation de chaque critère en trois zones, est défini à partir d'un seuil minimum et un seuil maximum. Les valeurs inférieures au seuil minimum apparaissent en caractères gras soulignés, les valeurs supérieures au seuil maximum apparaissent en caractères gras surlignés, et les valeurs comprises entre les deux seuils sont affichées dans la police standard. La colonne intitulée $M\log(J!)$ donne une estimation du logarithme de la taille de l'espace de recherche (voir 7.2.1, page 91). La colonne intitulée $\%_{\mathsf{opt}}$ contient l'écart (en %) de la moyenne du makespan des solutions finales de l'AIRLNDM(f_3) à la valeur du makespan optimum (ou à sa borne inférieure si ce dernier n'est pas connu).

$\%_{\mathsf{opt}}$	$M\log(J!)$	Inst	JxM	Opt	$C^l_{\max}$	$\pi_{\max}+r_{\max}$	$C^u_{\max}$	ρ-algo.	moy	Δ_r	$u_r\%$	$\frac{p}{\Pi}\%$	$\sigma_\%$	$\sigma_{mr\%}$	$\sigma_{jr\%}$	$\sigma_{tr\%0}$	$\sigma_{ur\%0}$	$\overline{D_r}\%$	$D_{wr\%}$
	13	dd00	4x4	272	186	369	671	$\rho_3=3.9$	**42**	**12**	**227**	**99**	**71**	**25**	35	**44**	**18**	**17**	59
5.22	39	ft06	6x6	55	47	90	154	$\rho_7=3.7$	**6**	**19**	**176**	**110**	**75**	**19**	47	**41**	36	**12**	**23**
11.77	60	car7	7x7	6558	4216	7865	23884	$\rho_7=6.0$	**488**	**41**	198	**116**	**57**	10	46	**10**	**19**	**100**	**0**
0.02	76	la05	10x5	593	593	973	2283	$\rho_7=4.8$	**46**	**38**	213	65	56	14	**50**	23	49	**20**	52
3.45	76	la01	10x5	666	666	1079	2849	$\rho_7=4.9$	**57**	**39**	**172**	63	45	13	**26**	**11**	51	**20**	53
6.88	76	la04	10x5	590	537	906	2507	$\rho_7=4.9$	51	**40**	196	69	**64**	12	**31**	**8**	51	**22**	49
8.98	76	la03	10x5	597	588	937	2383	$\rho_7=4.8$	**48**	**38**	191	60	53	13	**56**	15	50	**29**	48
9.40	76	la02	10x5	655	635	1029	2643	$\rho_7=4.9$	53	**38**	188	63	51	10	**32**	**14**	**57**	**20**	51
12.14	85	car8	8x8	8264	4908	9521	31883	$\rho_7=6.0$	**499**	**55**	194	**94**	51	15	**33**	19	**15**	**100**	**0**
15.06	88	car1	11x5	7038	6143	9231	25025	$\rho_7=6.0$	**455**	**42**	**220**	**51**	**60**	**6**	35	**13**	43	**100**	**0**
15.14	90	car2	13x4	7166	6828	9675	23715	$\rho_3=3.9$	**457**	**38**	**219**	**42**	**60**	11	**48**	**8**	**74**	**100**	**0**
11.27	91	car5	10x6	7702	5912	9949	29452	$\rho_7=6.0$	**491**	**48**	204	69	**60**	**9**	**32**	**14**	36	**100**	**0**
16.32	100	car3	12x5	7312	6018	9466	27984	$\rho_7=6.0$	**467**	**48**	208	58	**58**	**8**	42	**6**	47	**100**	**0**
12.08	101	car4	14x4	8003	7212	10183	26054	$\rho_3=3.9$	**466**	**41**	**215**	**42**	**61**	11	35	**7**	**62**	**100**	**0**
17.51	102	car6	8x9	8313	5486	10387	34819	$\rho_7=6.0$	**484**	**61**	207	**112**	**61**	**7**	**32**	19	**14**	**100**	**0**
0.00	139	la06	15x5	926	926	1339	3992	$\rho_7=4.9$	54	**58**	185	**45**	50	14	45	**6**	50	**18**	54
0.01	139	la10	15x5	958	958	1401	4020	$\rho_7=4.8$	54	**58**	181	**47**	53	10	**58**	**9**	**58**	**18**	53
0.06	139	la09	15x5	951	951	1333	4263	$\rho_7=4.9$	**57**	**59**	**175**	**41**	47	**8**	46	**7**	46	**20**	53
1.02	139	la08	15x5	863	863	1232	3825	$\rho_7=4.9$	51	**59**	193	**43**	**60**	**8**	35	**6**	51	**20**	53
1.56	139	la07	15x5	890	869	1245	3745	$\rho_7=4.8$	50	**58**	195	**44**	45	**8**	36	**7**	45	**22**	51
7.59	151	la17	10x10	784	683	1329	4676	$\rho_7=7.3$	**47**	**86**	210	**95**	**59**	**17**	**55**	25	**18**	**8**	59
7.64	151	abz6	10x10	943	742	1430	5946	$\rho_7=7.3$	**60**	**88**	**165**	**108**	39	14	36	**12**	**15**	**12**	58
7.83	151	la16	10x10	945	717	1377	5351	$\rho_7=7.3$	54	**87**	184	**109**	50	14	**33**	**14**	**19**	**11**	55
8.45	151	la20	10x10	902	756	1500	5445	$\rho_7=7.3$	55	**87**	182	**102**	51	**20**	**34**	15	**19**	**12**	57
8.67	151	abz5	10x10	1234	868	1727	7773	$\rho_7=7.3$	**78**	**89**	**128**	**99**	**19**	**17**	**24**	**6**	**6**	**10**	59
9.23	151	la18	10x10	848	663	1286	5186	$\rho_7=7.3$	52	**88**	188	**107**	50	14	37	17	**16**	**9**	57
10.63	151	la19	10x10	842	685	1302	5346	$\rho_7=7.3$	54	**88**	182	**91**	53	15	44	17	**10**	**11**	**60**
11.24	151	orb9	10x10	934	661	1320	5181	$\rho_7=7.3$	52	**88**	192	**100**	55	**17**	44	15	**21**	**16**	43
12.11	151	orb2	10x10	888	671	1291	5255	$\rho_7=7.3$	53	**88**	189	**93**	55	**18**	47	18	**16**	**15**	52
12.24	151	orb4	10x10	1005	759	1512	5628	$\rho_7=7.3$	**57**	**87**	**175**	**100**	47	**21**	**30**	15	**19**	**16**	43
13.53	151	orb7	10x10	397	286	561	2407	$\rho_7=6.9$	**25**	**89**	**246**	**97**	48	**20**	**53**	**12**	**10**	**15**	52
14.73	151	ft10	10x10	930	655	1286	5109	$\rho_7=7.3$	52	**88**	194	**104**	54	14	41	**13**	**17**	**16**	38
15.45	151	orb5	10x10	887	630	1214	4908	$\rho_7=7.3$	50	**88**	202	**93**	56	15	37	17	**12**	**27**	39
16.53	151	orb1	10x10	1059	695	1338	5409	$\rho_7=7.3$	55	**88**	184	**109**	53	**17**	41	**12**	**16**	**20**	42
16.91	151	orb8	10x10	899	585	1158	4560	$\rho_7=7.3$	**46**	**88**	213	**98**	**62**	**16**	**50**	15	**13**	**26**	31

Étude des instances de JSP : valeur de quelques critères .../...

$\%_{\text{opt}}$	$M\log(J)$	Inst	JxM	Opt	$C^l_{\max}$	$\pi_{\max}+r_{\max}$	$C^u_{\max}$	ρ-algo.	moy	Δ_r	$u_{r\%}$	$\frac{p_{\mathcal{C}}}{\Pi}\%$	$\sigma_{r\%}$	$\sigma_{mr\%}$	$\sigma_{jr\%}$	$\sigma_{tr\%0}$	$\sigma_{ur\%0}$	$\overline{D_{r\%}}$	$D_{wr\%}$
17.33	151	orb6	10x10	1010	715	1374	5614	$\rho_7=7.3$	**57**	**88**	177	**109**	48	10	**32**	**10**	**15**	**20**	42
18.33	151	orb10	10x10	944	681	1333	5549	$\rho_7=7.3$	**56**	**88**	179	**105**	51	10	44	**14**	**12**	**27**	39
21.19	151	orb3	10x10	1005	648	1272	5292	$\rho_7=7.3$	53	**88**	188	**104**	54	15	**32**	**11**	**14**	**26**	31
0.00	212	la11	20x5	1222	1222	1635	5351	$\rho_7=4.9$	54	**78**	184	**34**	50	10	40	**5**	51	**20**	51
0.00	212	la12	20x5	1039	1039	1447	4676	$\rho_7=4.9$	**47**	**78**	210	**40**	**59**	10	**58**	**7**	**68**	**20**	51
0.00	212	la14	20x5	1292	1292	1735	5346	$\rho_7=4.8$	54	**76**	182	**35**	53	**5**	44	**7**	**55**	**18**	54
0.09	212	la13	20x5	1150	1150	1532	5186	$\rho_7=4.8$	52	**78**	188	**34**	50	12	**50**	**6**	**54**	**20**	53
3.15	212	la15	20x5	1207	1207	1585	5445	$\rho_7=4.9$	55	**78**	182	**32**	51	13	**50**	**4**	46	**21**	50
13.73	212	ft20	20x5	1165	1119	1506	5109	$\rho_7=4.9$	52	**79**	194	**35**	54	**9**	**49**	**4**	**52**	**32**	43
8.24	279	la23	15x10	1032	1032	1672	8085	$\rho_7=7.3$	54	**131**	180	63	51	**17**	41	**9**	**11**	**10**	**60**
14.02	279	la25	15x10	977	864	1587	7509	$\rho_7=7.3$	51	**133**	198	84	56	12	44	**6**	**22**	**11**	58
14.05	279	la21	15x10	1046	935	1652	7994	$\rho_7=7.3$	54	**133**	186	77	50	**9**	40	**8**	**18**	**11**	58
14.97	279	la24	15x10	935	857	1561	7727	$\rho_7=7.3$	52	**134**	193	83	53	15	44	**8**	**17**	**10**	**60**
16.50	279	la22	15x10	927	830	1449	7322	$\rho_7=7.3$	49	**133**	201	75	**58**	13	41	**8**	**16**	**11**	56
13.56	418	la36	15x15	1268	1028	1976	11739	$\rho_7=7.3$	53	**206**	190	**93**	49	14	**50**	**9**	**10**	**6**	**72**
14.86	418	ta06	15x15	1238	889	1738	11351	$\rho_7=7.3$	51	**208**	197	**96**	**57**	**9**	44	**9**	**7**	**7**	**72**
15.32	418	ta07	15x15	1227	935	1855	11553	$\rho_7=7.3$	52	**207**	193	**102**	**57**	**16**	39	**9**	**9**	**8**	**72**
15.53	418	la37	15x15	1397	986	1966	12569	$\rho_7=7.3$	**56**	**208**	178	**101**	49	13	**52**	**9**	**7**	**8**	**71**
15.59	418	ta02	15x15	1244	942	1861	11356	$\rho_7=7.3$	51	**207**	197	**103**	56	14	47	**9**	**10**	**7**	**72**
16.60	418	ta10	15x15	1241	911	1807	11356	$\rho_7=7.3$	51	**207**	197	**99**	**59**	11	44	**10**	**8**	**7**	**72**
16.71	418	la39	15x15	1233	1012	1934	11554	$\rho_7=7.3$	52	**206**	193	**92**	52	**17**	46	**11**	**9**	**7**	**73**
16.98	418	ta01	15x15	1231	977	1940	11671	$\rho_7=7.3$	52	**207**	191	**99**	**57**	**9**	38	**10**	**9**	**7**	**74**
17.16	418	ta03	15x15	1218	921	1821	11181	$\rho_7=7.3$	50	**207**	200	**103**	**58**	14	40	**9**	**10**	**7**	**72**
17.18	418	la40	15x15	1222	1027	1982	11472	$\rho_7=7.3$	51	**205**	195	**93**	54	13	47	**12**	**8**	**9**	**70**
17.57	418	ta05	15x15	1224	940	1842	11252	$\rho_7=7.3$	51	**207**	198	**105**	**58**	**9**	**51**	**9**	**9**	**8**	**71**
17.75	418	ta08	15x15	1217	963	1823	11402	$\rho_7=7.3$	51	**206**	196	**112**	**62**	**9**	**55**	**7**	**13**	**7**	**74**
18.29	418	ta09	15x15	1274	982	1948	11835	$\rho_7=7.3$	53	**207**	189	**102**	54	11	38	**10**	**9**	**7**	**69**
18.30	418	ta04	15x15	1175	911	1781	10949	$\rho_7=7.3$	49	**207**	202	**105**	**57**	**16**	43	**11**	**8**	**7**	**72**
19.48	418	la38	15x15	1196	943	1819	11217	$\rho_7=7.3$	50	**207**	199	**108**	56	10	42	**8**	**11**	**6**	**73**
9.59	423	la30	20x10	1355	1355	2081	10680	$\rho_7=7.3$	54	**175**	186	54	53	10	44	**6**	**25**	**10**	58
12.32	423	la26	20x10	1218	1218	1935	10515	$\rho_7=7.3$	53	**177**	189	59	50	**9**	**51**	**5**	**18**	**10**	58
13.98	423	la28	20x10	1216	1216	1972	10682	$\rho_7=7.3$	54	**178**	186	63	51	**6**	47	**4**	**17**	**11**	59
14.90	423	la27	20x10	1235	1188	1874	10832	$\rho_7=7.3$	55	**179**	183	58	49	**8**	43	**4**	**14**	**11**	58
18.04	423	la29	20x10	$\frac{1142}{1153}$	1105	1828	9929	$\rho_7=7.3$	50	**178**	200	66	56	**8**	**53**	**3**	**20**	**10**	58
17.10	635	ta14	20x15	1345	1130	2120	14310	$\rho_7=7.3$	**48**	**277**	208	**88**	**58**	12	**53**	**7**	**9**	**6**	**72**
17.91	635	abz7	20x15	656	556	966	7366	$\rho_3=5.3$	**25**	**278**	**163**	74	**34**	14	**49**	**5**	**5**	**7**	**73**
19.41	635	ta17	20x15	$\frac{1458}{1464}$	1257	2236	15544	$\rho_7=7.3$	52	**276**	192	78	55	12	**50**	**7**	**9**	**7**	**73**
22.43	635	ta18	20x15	$\frac{1369}{1396}$	1153	2053	14919	$\rho_7=7.3$	50	**277**	200	79	**58**	12	**51**	**5**	**8**	**7**	**71**
23.63	635	ta20	20x15	$\frac{1316}{1353}$	1186	2114	15133	$\rho_7=7.3$	51	**277**	197	79	**58**	11	43	**6**	**9**	**7**	**72**
23.81	635	abz8	20x15	$\frac{645}{669}$	566	1009	7586	$\rho_3=5.3$	**26**	**278**	**159**	79	**35**	10	39	**4**	**7**	**8**	**71**
24.98	635	abz9	20x15	$\frac{661}{679}$	563	1030	7442	$\rho_3=5.3$	**25**	**278**	**162**	83	**36**	**9**	38	**4**	**10**	**8**	**72**
25.44	635	ta11	20x15	$\frac{1321}{1364}$	1139	2088	14447	$\rho_7=7.3$	49	**277**	206	84	**58**	10	**54**	**6**	**12**	**8**	**71**
26.08	635	ta16	20x15	$\frac{1300}{1362}$	1181	2113	15540	$\rho_7=7.3$	52	**278**	190	79	55	14	**54**	**6**	**10**	**7**	**72**
26.19	635	ta12	20x15	$\frac{1321}{1367}$	1251	2263	15559	$\rho_7=7.3$	52	**276**	191	81	55	15	47	**6**	**9**	**7**	**72**
26.80	635	ta19	20x15	$\frac{1276}{1341}$	1202	2122	14773	$\rho_7=7.3$	50	**276**	202	77	**59**	**8**	**52**	**7**	**9**	**8**	**70**
26.99	635	ta13	20x15	$\frac{1271}{1350}$	1178	2097	14541	$\rho_7=7.3$	49	**276**	205	79	**62**	10	42	**6**	**11**	**6**	**73**
27.76	635	ta15	20x15	$\frac{1293}{1342}$	1148	2028	14591	$\rho_7=7.3$	49	**277**	204	77	**60**	10	43	**6**	**10**	**7**	**72**
2.50	747	la33	30x10	1719	1719	2442	14969	$\rho_7=7.3$	50	**266**	199	**43**	55	**7**	**48**	**3**	**18**	**10**	58
2.86	747	la31	30x10	1784	1784	2501	15191	$\rho_7=7.3$	51	**265**	196	**41**	53	**7**	45	**4**	**19**	**11**	58
3.78	747	la32	30x10	1850	1850	2606	16569	$\rho_7=7.3$	**56**	**267**	180	**41**	50	**7**	44	**2**	**15**	**11**	58
5.67	747	la35	30x10	1888	1888	2535	15485	$\rho_7=7.3$	52	**264**	192	**35**	54	10	45	**5**	**16**	**11**	58
5.98	747	la34	30x10	1721	1721	2377	15341	$\rho_7=7.3$	52	**267**	194	**39**	55	**9**	**48**	**4**	**13**	**10**	59

Étude des instances de JSP : valeur de quelques critères .../...

$\%_{\text{opt}}$	$M\log(J)$	Inst	JxM	Opt	$C^l_{\max}$	$\Pi_{\max}+P_{\max}$	$C^u_{\max}$	ρ-algo.	moy	Δ_r	$u_{r\%}$	$\frac{p_Q}{\Pi}\%$	$\sigma_{r\%}$	$\sigma_{mr\%}$	$\sigma_{jr\%}$	$\sigma_{tr\%0}$	$\sigma_{ur\%0}$	$\overline{D_{r\%}}$	$D_{wr\%}$
20.79	847	ta24	20x20	$\frac{1602}{1651}$	1271	2422	20400	$\rho_7=7.3$	51	**376**	195	**91**	**57**	10	43	**7**	**5**	**6**	**88**
22.38	847	ta28	20x20	$\frac{1591}{1615}$	1269	2490	20202	$\rho_7=7.3$	51	**375**	197	**97**	**59**	10	43	**8**	**8**	**5**	**89**
27.19	847	ta25	20x20	$\frac{1504}{1597}$	1256	2426	19654	$\rho_7=7.3$	50	**375**	202	**94**	**59**	10	**48**	**8**	**7**	**6**	**87**
27.85	847	ta27	20x20	$\frac{1616}{1687}$	1331	2622	21091	$\rho_7=7.3$	53	**375**	188	**97**	55	**7**	**52**	**8**	**6**	**6**	**87**
28.19	847	ta23	20x20	$\frac{1472}{1558}$	1185	2349	20086	$\rho_7=7.3$	51	**377**	198	**99**	**57**	11	43	**7**	**5**	**5**	**88**
28.46	847	ta21	20x20	$\frac{1539}{1645}$	1217	2399	20169	$\rho_7=7.3$	51	**376**	197	**103**	**59**	10	**50**	**6**	**7**	**5**	**88**
28.86	847	ta22	20x20	$\frac{1511}{1601}$	1240	2463	20143	$\rho_7=7.3$	51	**376**	197	**99**	**59**	**7**	42	**6**	**6**	**6**	**88**
28.98	847	ta26	20x20	$\frac{1539}{1651}$	1207	2412	20593	$\rho_7=7.3$	52	**377**	193	**101**	54	11	46	**6**	**6**	**6**	**88**
30.05	847	ta29	20x20	$\frac{1514}{1625}$	1267	2494	20982	$\rho_7=7.3$	53	**376**	189	**97**	54	12	43	**7**	**6**	**6**	**88**
32.18	847	ta30	20x20	$\frac{1473}{1585}$	1212	2371	20052	$\rho_7=7.3$	51	**376**	198	**105**	**58**	10	**55**	**6**	**7**	**6**	**87**
9.82	1120	ta35	30x15	2007	1729	2698	22433	$\rho_7=7.3$	50	**416**	199	57	**59**	**8**	**54**	**3**	**11**	**7**	**72**
18.66	1120	ta39	30x15	1795	1641	2546	21583	$\rho_7=7.3$	**48**	**416**	207	56	**59**	**7**	**51**	**3**	**7**	**7**	**71**
20.02	1120	ta34	30x15	$\frac{1828}{1832}$	1828	2803	23443	$\rho_7=7.3$	53	**415**	191	54	55	**8**	47	**3**	**9**	**7**	**73**
20.52	1120	ta31	30x15	1764	1764	2754	22600	$\rho_7=7.3$	51	**415**	198	57	**58**	13	**52**	**4**	**11**	**7**	**72**
22.59	1120	ta36	30x15	1819	1777	2765	22959	$\rho_7=7.3$	52	**416**	195	56	**57**	10	46	**4**	**11**	**7**	**72**
23.04	1120	ta37	30x15	$\frac{1771}{1784}$	1771	2816	23807	$\rho_7=7.3$	53	**417**	188	60	54	**9**	**53**	**3**	**10**	**7**	**73**
23.49	1120	ta38	30x15	$\frac{1673}{1677}$	1673	2625	22323	$\rho_7=7.3$	50	**417**	200	57	**57**	**9**	**49**	**3**	**9**	**7**	**73**
24.41	1120	ta32	30x15	$\frac{1774}{1803}$	1774	2746	23206	$\rho_7=7.3$	52	**416**	192	55	53	**7**	**52**	**3**	**11**	**7**	**72**
24.97	1120	ta33	30x15	$\frac{1778}{1796}$	1729	2785	23693	$\rho_7=7.3$	53	**418**	189	62	54	**9**	47	**3**	**8**	**7**	**72**
26.85	1120	ta40	30x15	$\frac{1631}{1686}$	1602	2563	22310	$\rho_7=7.3$	50	**418**	200	60	**59**	10	**49**	**3**	**10**	**7**	**73**
23.64	1493	ta45	30x20	$\frac{1997}{2005}$	1731	2984	30646	$\rho_7=7.3$	52	**567**	194	73	56	10	**50**	**3**	**6**	**6**	**88**
25.58	1493	ta48	30x20	$\frac{1912}{1971}$	1744	3026	29972	$\rho_7=7.3$	50	**566**	199	74	56	**8**	**54**	**4**	**6**	**5**	**88**
28.04	1493	ta49	30x20	$\frac{1915}{1984}$	1758	2948	29752	$\rho_7=7.3$	50	**565**	200	68	**57**	11	**48**	**4**	**6**	**6**	**88**
28.30	1493	ta46	30x20	$\frac{1940}{2029}$	1856	3146	30823	$\rho_7=7.3$	52	**564**	193	70	54	**9**	**51**	**3**	**7**	**6**	**87**
29.16	1493	ta44	30x20	$\frac{1927}{1998}$	1787	2991	30486	$\rho_7=7.3$	51	**565**	195	68	**57**	11	**49**	**4**	**6**	**6**	**88**
31.28	1493	ta42	30x20	$\frac{1867}{1961}$	1761	2953	30591	$\rho_7=7.3$	51	**566**	195	68	**57**	**8**	**48**	**3**	**5**	**6**	**89**
31.90	1493	ta43	30x20	$\frac{1809}{1879}$	1694	2924	29578	$\rho_7=7.3$	50	**566**	201	73	**59**	**7**	**56**	**3**	**7**	**6**	**88**
33.04	1493	ta47	30x20	$\frac{1789}{1913}$	1690	3024	30223	$\rho_7=7.3$	51	**567**	197	79	**57**	10	**48**	**3**	**8**	**5**	**89**
36.15	1493	ta41	30x20	$\frac{1859}{2023}$	1830	3062	31279	$\rho_7=7.3$	53	**565**	190	68	55	12	**51**	**4**	**6**	**5**	**87**
36.69	1493	ta50	30x20	$\frac{1807}{1937}$	1674	2925	30657	$\rho_7=7.3$	52	**568**	194	75	56	**9**	**48**	**3**	**6**	**5**	**89**
19.71	2227	ta51	50x15	2760	2760	3735	37918	$\rho_7=7.3$	51	696	196	**36**	56	**8**	**55**	**2**	**9**	**7**	**73**
	2227	ta52	50x15	2756	2756	3763	37144	$\rho_7=7.3$	50	695	200	**37**	**60**	**6**	**55**	**2**	**10**	**7**	**72**
	2227	ta53	50x15	2717	2717	3659	37381	$\rho_7=7.3$	50	696	199	**35**	**57**	**7**	**57**	**2**	**9**	**7**	**73**
	2227	ta54	50x15	2839	2797	3941	36991	$\rho_7=7.3$	50	694	201	**41**	**58**	**6**	**52**	**2**	**10**	**7**	**72**
	2227	ta55	50x15	2679	2679	3619	36869	$\rho_7=7.3$	50	696	202	**36**	**58**	**7**	**50**	**2**	**9**	**7**	**73**
	2227	ta56	50x15	2781	2781	3701	37948	$\rho_7=7.3$	51	696	196	**34**	**57**	**9**	**54**	**2**	**10**	**7**	**72**
	2227	ta57	50x15	2943	2943	4080	38432	$\rho_7=7.3$	52	693	194	**39**	**57**	**6**	**53**	**2**	**11**	**7**	**73**
	2227	ta58	50x15	2885	2885	3927	39085	$\rho_7=7.3$	53	695	190	**37**	55	**7**	**54**	**2**	**9**	**7**	**72**
	2227	ta59	50x15	2655	2655	3618	36864	$\rho_7=7.3$	50	696	202	**37**	**58**	**7**	**52**	**2**	**10**	**7**	**73**
	2227	ta60	50x15	2723	2723	3784	37796	$\rho_7=7.3$	51	696	197	**39**	**58**	**5**	**53**	**2**	**11**	**7**	**73**
19.25	2970	ta61	50x20	2868	2868	4152	50964	$\rho_7=7.3$	51	944	195	**45**	56	**5**	**51**	**2**	**6**	**5**	**88**
	2970	ta62	50x20	$\frac{2869}{2872}$	2848	4166	51598	$\rho_7=7.3$	52	945	192	**47**	56	**7**	**51**	**2**	**8**	**6**	**88**
	2970	ta63	50x20	2755	2755	4044	49121	$\rho_7=7.3$	50	944	202	**47**	**59**	**8**	**54**	**2**	**7**	**5**	**88**
	2970	ta64	50x20	2702	2691	3920	48462	$\rho_7=7.3$	49	945	205	**46**	**59**	**6**	**56**	**2**	**8**	**5**	**89**
	2970	ta65	50x20	2725	2725	3995	49199	$\rho_7=7.3$	50	945	202	**47**	**60**	**6**	**52**	**2**	**6**	**5**	**88**
	2970	ta66	50x20	2845	2845	4142	50390	$\rho_7=7.3$	51	944	197	**46**	**58**	**7**	**51**	**2**	**7**	**5**	**88**
	2970	ta67	50x20	2825	2812	4085	49553	$\rho_7=7.3$	50	944	200	**46**	**58**	**6**	**53**	**2**	**7**	**6**	**88**
	2970	ta68	50x20	2784	2764	4107	49903	$\rho_7=7.3$	50	945	199	**49**	**57**	**8**	**54**	**2**	**7**	**6**	**89**
	2970	ta69	50x20	3071	3063	4479	50871	$\rho_7=7.3$	51	940	195	**47**	**57**	**7**	**55**	**2**	**7**	**5**	**89**
	2970	ta70	50x20	2995	2995	4220	50878	$\rho_7=7.3$	51	942	195	**41**	**57**	**6**	**50**	**2**	**6**	**6**	**88**
10.52	7275	ta71	100x20	5464	5464	6805	100891	$\rho_4=6.3$	51	**1892**	197	**25**	**57**	**5**	**55**	**1**	**7**	**6**	**88**
	7275	ta72	100x20	5181	5181	6456	97365	$\rho_4=6.3$	49	**1894**	204	**25**	**59**	**5**	**54**	**1**	**7**	**5**	**88**

Étude des instances de JSP : valeur de quelques critères .../...

$\%_{\text{opt}}$	$M\log(J)$	Inst	JxM	Opt	$C^l_{\max}$	$\Pi_{\max}+P_{\max}$	$C^u_{\max}$	ρ-algo.	moy	Δ_r	$u_{r\%}$	$\frac{p}{\Pi}\%$	$\sigma_{r\%}$	$\sigma_{nr\%}$	$\sigma_{jr\%}$	$\sigma_{tr\%0}$	$\sigma_{ur\%0}$	$\overline{D_{r\%}}$	$D_{wr\%}$
7275		ta73	100x20	5568	5552	6858	100622	$\rho_4=6.3$	51	**1890**	197	**24**	**57**	**4**	**55**	**1**	**7**	**6**	**88**
7275		ta74	100x20	5339	5339	6636	100049	$\rho_4=6.3$	51	**1894**	198	**25**	**57**	**4**	**56**	**1**	**7**	**5**	**88**
7275		ta75	100x20	5392	5392	6739	98720	$\rho_4=6.3$	50	**1891**	201	**25**	**58**	**4**	**55**	**1**	**7**	**6**	**88**
7275		ta76	100x20	5342	5342	6612	100371	$\rho_4=6.3$	51	**1894**	198	**24**	**57**	**5**	**56**	**1**	**6**	**6**	**88**
7275		ta77	100x20	5436	5436	6680	101716	$\rho_4=6.3$	51	**1894**	195	**23**	**57**	**4**	**55**	**1**	**7**	**5**	**88**
7275		ta78	100x20	5394	5394	6693	99100	$\rho_4=6.3$	50	**1892**	200	**25**	**58**	**5**	**57**	**1**	**7**	**6**	**88**
7275		ta79	100x20	5358	5358	6660	99733	$\rho_4=6.3$	50	**1893**	199	**25**	**58**	**6**	**57**	**1**	**6**	**6**	**88**
7275		ta80	100x20	5183	5183	6409	96697	$\rho_4=6.3$	49	**1893**	205	**24**	**59**	**4**	**57**	**1**	**7**	**6**	**88**

Tableau A.6: Étude des instances de JSP : valeur de quelques critères

A.3 Résultats détaillés des schémas d'application

Cette section rassemble les résultats obtenus par notre algorithme évolutif en utilisant les schémas d'application parallèle et séquentiel. Plus précisément, l'algorithme évolutif utilisé est une version modifiée de *LibGA* (voir section D.1, page 223). Le paramétrage utilisé est le suivant :

- codage : direct ;
- fonction coût : exclusivement basée sur le makespan (f_1) ;
- type d'algorithme évolutif : AG générationnel, élitiste ;
- taux de remplacement : 100% ;
- type de sélection : SUS sans doublon (biais de 1.25) ;
- crossover : GA/GT (taux de mutation interne fixé à 0) ;
- mutation : invert1, invert2 ou invert6 ;
- taux de crossover : 0.6 ou 0.9 selon expérience ;
- taux de mutation : 0.6 ou 0.9 selon expérience ;
- critère d'arrêt : nombre de générations ;
- taille de population : variable, selon expérience ;
- nombre de générations : variable, selon expérience.

La sélection effectuée à l'issue du crossover dans le schéma d'application parallèle est déterministe : elle choisit systématiquement le meilleur individu.

Pour chaque instance i, un ensemble Π_i de 10 populations initiales P_i^j est constitué : $\Pi_i = \{P_i^j \ 1 \leq j \leq 10\}$. Chaque paramétrage p est ensuite testé à partir de Π_i ce qui permet de comparer précisément l'impact du paramétrage sur la qualité des solutions finales, ou sur les diverses mesures effectuées après chaque génération. De cette façon, pour chaque instance i et chaque paramétrage p nous obtenons un ensemble de 10 populations finales $\Pi'_i = \{P_i'^j \ 1 \leq j \leq 10\}$:

$$\Pi_i \xrightarrow{\mathsf{AE}(p)} \Pi'_i$$

Nous définissons ensuite cinq sous-ensembles de Π'_i relatifs à cinq critères d'étude :

1. ensemble des meilleures solutions des populations finales $\Pi'_{\mathsf{min}} = \{\min_j P_i'^j\}$
2. ensemble des plus mauvaises solutions des populations finales $\Pi'_{\mathsf{max}} = \{\max_j P_i'^j\}$

3. ensemble des valeurs moyennes des populations finales $\Pi'_{\mathsf{moy}} = \{\text{moy}_j P_i'^j\}$
4. ensemble des valeurs des écarts-types des populations finales $\Pi'_{\mathsf{écart}} = \{\text{écart}_j P_i'^j\}$
5. ensemble des valeurs relatives au nombre d'évaluations $\Pi'_{\mathsf{éval}} = \{\text{éval}_j P_i'^j\}$

Pour chaque critère, nous adoptons les conventions suivantes pour présenter les résultats sous forme de tableau :

- sch représente le schéma d'application utilisé : 1 pour le schéma séquentiel, 2 pour le schéma parallèle ;
- taux représente le taux d'application des opérateurs : 0906 correspond à un taux de mutation de 0.9 et un taux de crossover de 0.6
- moy, écart, max et min représentent respectivement la moyenne, l'écart-type, la valeur maximale, la valeur minimale du critère étudié sur les populations de Π'_i.

Pour chaque instance et chaque paramétrage :

1. la colonne min du tableau relatif à Π'_{min} donne la meilleure performance obtenue (*i.e.* le makespan minimum sur les dix expériences) ;
2. la colonne max du tableau relatif à Π'_{max} donne la plus mauvaise performance obtenue (*i.e.* le makespan maximum sur les dix expériences) ;
3. la colonne moy du tableau relatif à Π'_{moy} donne la performance moyenne sur les dix expériences ;
4. la colonne écart du tableau relatif à $\Pi'_{\mathsf{écart}}$ permet d'étudier la dispersion (en terme d'écart-type) des solutions en moyenne sur les dix expériences ;
5. les colonnes min et max du tableau relatif à Π'_{moy} donnent respectivement la meilleure et la plus mauvaise performance obtenue en considérant la moyenne des makespan de chaque population (en quelque sorte la meilleure et la plus mauvaise population obtenue en considérant leur performance moyenne) ;
6. la colonne écart du tableau relatif à Π'_{moy} fournit un indicateur sur la disparité moyenne des populations obtenues (indique si la moyenne sur chaque population varie d'une expérience à l'autre) ;
7. les colonnes moy, écart, max et min du tableau relatif à $\Pi'_{\mathsf{éval}}$ représentent respectivement la moyenne, l'écart-type, la valeur maximale, la valeur minimale du nombre d'évaluations nécessaires pour engendrer les populations de Π'_i.

Les quatre premières mesures correspondent aux mesures classiquement effectuées (minimum, maximum, moyenne, écart-type) ; mais il est possible de combiner différemment les critères pour mettre en évidence telle ou telle particularité d'un paramétrage ou d'une instance.

Nous disposons d'un grand nombre d'expériences préliminaires, non présentées dans ce mémoire : elles ont été utilisées pour tester différents paramétrages de l'algorithme évolutif. Les résultats obtenus au cours de ces expériences confirment les résultats présentés dans cette section.

Nous introduisons un certain nombre de notations pour faciliter la synthèse des résultats. Tout d'abord le titre des sous-sections précise la taille de population (en *nombre d'individus*) ainsi que le

nombre de générations effectuées. Par défaut, lorsqu'aucune précision n'est donnée, la fonction coût utilisée est f_1 (coût exclusivement lié au makespan).

Lors de l'utilisation d'une fonction multicritère pour guider l'AE, nous calculons un certain nombre de statistiques liées au makespan afin de comparer le gain en terme de makespan lors de l'utilisation des fonctions multicritères.

Le paramétrage (500 individus, 300 générations) constitue un compromis entre nombre d'évaluations et qualité de solution finale, mais il est clair qu'il ne permet pas d'obtenir les meilleures performances possibles pour l'AE ou le meilleur gain entre schéma séquentiel et schéma parallèle.

Nous nous limitons à la présentation de quelques résultats dans le cadre de cette section afin de ne pas allonger démesurément les tableaux de résultats. Bien que nous nous soyons concentrés sur une instance (ta01) pour illustrer nos travaux, des résultats similaires ont été obtenus pour d'autres instances ou d'autres paramétrages.

A.3.1 Durée d'exécution – ordre de grandeur

La durée d'exécution d'un algorithme ne fait pas partie des critères sur lesquels nous insistons car elle dépend de nombreux paramètres qui risquent de biaiser les comparaisons (version du système d'exploitation, version du compilateur, code optimisé ou version quasi-canonique, dans certains cas la charge du calculateur utilisé, ...). Pour ces diverses raisons, nous donnons juste une estimation du temps d'exécution de l'AE guidé par f_1 pour une instance ta01, après 300 générations de 500 individus (taux de mutation et de crossover fixés à 0.9) en schéma d'application parallèle : $\approx$ 9 minutes. L'ordinateur utilisé est un pentium-pro cadencé à 200MHz doté de 64Mo de mémoire EDO/60ns fonctionnant sous LINUX1.2.13. Partant d'un code non optimisé, sur un ordinateur assimilé de nos jours à une pièce de musée, dans un paramétrage conduisant à un nombre d'évaluations maximum par génération ($\approx$ 203500 appels à la fonction coût pour 300 générations), nous pouvons raisonnablement considérer que 9 minutes constituent une borne supérieure largement surestimée si l'on considère un ordinateur récent. En définitive, bien que non optimal (en terme de code ou d'environnement d'exécution) l'algorithme évolutif utilisé induit des temps d'exécution tout à fait raisonnables compte tenu de la taille de l'instance (15x15) traitée.

Description des opérateurs de mutation

Les opérateurs de mutation utilisés sont nommés invert1, invert2, invert6. Ils fonctionnent tous selon le principe exposé en section 7.1.2 (page 82). La différence entre l'opérateur invert1 et l'opérateur invert6 réside dans la méthode de réordonnancement. L'opérateur invert1 autorise le réordonnancement des deux opérations permutées si cela apporte un gain, tandis que l'opérateur invert6 applique strictement les étapes décrites par le pseudo-algorithme 7.2 (page 82).

L'opérateur invert2 fonctionne sur le même principe que l'opérateur invert1 mais la sélection des deux opérations permutées est effectuée de manière à ce que l'une (au moins) des deux opérations soit une opération critique.

A.3.2 Résultats pour 500 individus après une génération

A.3.2.1 Critère Π'_{min}

inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	1203.50	2.41547e+01	1246	1173	ta01	1	0606	1	1550.50	1.97446e+01	1577	1506
ft10	1	0606	2	1171.70	3.89334e+01	1232	1106	ta01	1	0606	2	1604.80	2.73781e+01	1642	1551
ft10	2	0606	1	1198.60	2.89213e+01	1245	1150	ta01	2	0606	1	1542.20	1.70868e+01	1567	1508
ft10	2	0606	2	1198.90	1.85658e+01	1235	1164	ta01	2	0606	2	1595.80	4.22606e+01	1642	1520
ft10	6	0606	1	1198.30	4.98358e+01	1263	1118	ta01	6	0606	1	1569.60	7.96492e+00	1589	1559
ft10	6	0606	2	1208.50	2.33206e+01	1239	1156	ta01	6	0606	2	1601.00	3.57743e+01	1642	1545
ft10	1	0609	1	1223.00	3.00766e+01	1259	1150	ta01	1	0609	1	1540.40	2.05485e+01	1558	1498
ft10	1	0609	2	1216.00	3.18057e+01	1254	1159	ta01	1	0609	2	1595.40	2.95303e+01	1642	1552
ft10	2	0609	1	1247.20	2.97752e+01	1284	1196	ta01	2	0609	1	1545.70	1.52974e+01	1566	1523
ft10	2	0609	2	1202.90	1.82014e+01	1233	1175	ta01	2	0609	2	1596.30	2.84817e+01	1642	1548
ft10	6	0609	1	1231.80	2.35108e+01	1265	1176	ta01	6	0609	1	1566.60	1.21012e+01	1581	1546
ft10	6	0609	2	1206.70	2.95332e+01	1249	1170	ta01	6	0609	2	1607.00	3.78021e+01	1642	1526
ft10	1	0906	1	1184.00	3.17333e+01	1224	1106	ta01	1	0906	1	1550.20	9.46361e+00	1567	1537
ft10	1	0906	2	1185.40	2.71264e+01	1221	1125	ta01	1	0906	2	1583.00	3.58246e+01	1627	1487
ft10	2	0906	1	1197.50	2.85841e+01	1233	1127	ta01	2	0906	1	1529.30	1.52712e+01	1548	1499
ft10	2	0906	2	1197.00	3.75287e+01	1265	1132	ta01	2	0906	2	1575.10	3.64677e+01	1631	1516
ft10	6	0906	1	1209.80	2.56585e+01	1246	1163	ta01	6	0906	1	1563.90	9.99450e+00	1588	1551
ft10	6	0906	2	1185.20	2.32585e+01	1220	1151	ta01	6	0906	2	1611.50	1.44862e+01	1642	1587
ft10	1	0909	1	1204.40	2.39675e+01	1241	1166	ta01	1	0909	1	1543.40	1.11911e+01	1563	1524
ft10	1	0909	2	1196.30	2.44910e+01	1223	1142	ta01	1	0909	2	1608.80	2.39658e+01	1637	1567
ft10	2	0909	1	1207.40	3.59060e+01	1280	1163	ta01	2	0909	1	1541.30	1.66256e+01	1567	1505
ft10	2	0909	2	1182.20	2.57635e+01	1210	1132	ta01	2	0909	2	1586.30	2.10002e+01	1634	1563
ft10	6	0909	1	1230.30	1.77260e+01	1252	1206	ta01	6	0909	1	1550.80	1.77189e+01	1568	1522
ft10	6	0909	2	1196.30	2.55423e+01	1245	1164	ta01	6	0909	2	1606.50	2.36400e+01	1642	1565
ft20	1	0606	1	1450.20	2.78453e+01	1487	1394	abz7	1	0606	1	834.30	6.61891e+00	845	823
ft20	1	0606	2	1453.20	2.30035e+01	1491	1415	abz7	1	0606	2	861.60	7.43236e+00	871	843
ft20	2	0606	1	1453.50	2.42704e+01	1498	1415	abz7	2	0606	1	825.30	7.66877e+00	838	816
ft20	2	0606	2	1445.90	2.36916e+01	1483	1400	abz7	2	0606	2	860.20	8.98666e+00	875	842
ft20	6	0606	1	1444.50	2.25666e+01	1493	1415	abz7	6	0606	1	841.30	5.56866e+00	853	836
ft20	6	0606	2	1439.60	2.37874e+01	1477	1404	abz7	6	0606	2	863.60	1.09654e+01	877	840
ft20	1	0609	1	1467.00	2.83408e+01	1506	1415	abz7	1	0609	1	830.30	1.02083e+01	842	806
ft20	1	0609	2	1431.20	2.91301e+01	1474	1382	abz7	1	0609	2	865.00	8.50882e+00	873	844
ft20	2	0609	1	1462.90	3.26388e+01	1506	1410	abz7	2	0609	1	829.50	8.10247e+00	841	816
ft20	2	0609	2	1437.30	2.32983e+01	1492	1411	abz7	2	0609	2	866.00	1.13754e+01	877	840
ft20	6	0609	1	1468.30	2.69446e+01	1511	1415	abz7	6	0609	1	839.70	5.93380e+00	849	831
ft20	6	0609	2	1432.60	3.08940e+01	1474	1373	abz7	6	0609	2	865.20	1.17456e+01	877	838
ft20	1	0906	1	1433.80	3.02020e+01	1483	1378	abz7	1	0906	1	830.00	6.81175e+00	842	818
ft20	1	0906	2	1431.10	2.15195e+01	1459	1391	abz7	1	0906	2	850.20	1.22049e+01	868	830
ft20	2	0906	1	1427.00	2.09571e+01	1456	1388	abz7	2	0906	1	823.00	1.00100e+01	837	801
ft20	2	0906	2	1430.10	2.52248e+01	1469	1388	abz7	2	0906	2	855.60	8.01499e+00	867	841
ft20	6	0906	1	1434.60	2.43852e+01	1491	1402	abz7	6	0906	1	839.10	6.34744e+00	849	827
ft20	6	0906	2	1427.80	2.09370e+01	1459	1393	abz7	6	0906	2	865.60	7.96492e+00	877	852
ft20	1	0909	1	1449.30	2.24591e+01	1492	1415	abz7	1	0909	1	829.70	8.01311e+00	844	811
ft20	1	0909	2	1413.00	1.70880e+01	1432	1371	abz7	1	0909	2	849.00	1.01980e+01	863	833
ft20	2	0909	1	1456.10	2.24163e+01	1496	1415	abz7	2	0909	1	828.50	5.85235e+00	839	818
ft20	2	0909	2	1434.50	1.87417e+01	1462	1406	abz7	2	0909	2	853.20	1.64122e+01	875	823
ft20	6	0909	1	1458.00	3.56567e+01	1505	1383	abz7	6	0909	1	832.40	5.44426e+00	841	824
ft20	6	0909	2	1440.00	2.39291e+01	1461	1375	abz7	6	0909	2	864.70	1.21413e+01	877	839
inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min

A.3.2.2 Critère Π'_{max}

inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	1835.70	6.75989e+01	1959	1762	ta01	1	0606	1	2520.80	8.46236e+01	2655	2378
ft10	1	0606	2	1834.20	4.27991e+01	1903	1778	ta01	1	0606	2	2567.50	6.90395e+01	2704	2463
ft10	2	0606	1	1802.60	4.33502e+01	1872	1742	ta01	2	0606	1	2498.30	9.75869e+01	2720	2384
ft10	2	0606	2	1872.70	8.05705e+01	2048	1786	ta01	2	0606	2	2569.00	3.63896e+01	2632	2513
ft10	6	0606	1	1814.70	5.63685e+01	1904	1717	ta01	6	0606	1	2506.50	6.45202e+01	2594	2396
ft10	6	0606	2	1865.20	4.85321e+01	1949	1794	ta01	6	0606	2	2537.80	6.38244e+01	2656	2450
ft10	1	0609	1	1757.70	5.25834e+01	1904	1714	ta01	1	0609	1	2401.00	5.49509e+01	2527	2293
ft10	1	0609	2	1845.40	3.66857e+01	1917	1765	ta01	1	0609	2	2529.80	8.84520e+01	2720	2420
ft10	2	0609	1	1716.70	3.75634e+01	1777	1630	ta01	2	0609	1	2436.80	7.34967e+01	2553	2334
ft10	2	0609	2	1844.30	5.79276e+01	1964	1782	ta01	2	0609	2	2546.00	5.76715e+01	2652	2482
ft10	6	0609	1	1757.80	4.45417e+01	1848	1713	ta01	6	0609	1	2457.90	8.45641e+01	2628	2313
ft10	6	0609	2	1859.00	5.56758e+01	1970	1779	ta01	6	0609	2	2543.50	6.63027e+01	2662	2458
ft10	1	0906	1	1785.50	6.29702e+01	1911	1713	ta01	1	0906	1	2446.40	1.06333e+02	2720	2366
ft10	1	0906	2	1841.90	3.26541e+01	1892	1778	ta01	1	0906	2	2576.10	7.44157e+01	2696	2444
ft10	2	0906	1	1803.60	5.27678e+01	1905	1754	ta01	2	0906	1	2521.30	8.55009e+01	2689	2401
ft10	2	0906	2	1828.70	4.06769e+01	1882	1758	ta01	2	0906	2	2543.20	4.01841e+01	2643	2494
ft10	6	0906	1	1800.80	6.81011e+01	1954	1716	ta01	6	0906	1	2470.70	8.44287e+01	2626	2362
ft10	6	0906	2	1839.00	3.08448e+01	1892	1792	ta01	6	0906	2	2505.30	5.58284e+01	2633	2410
ft10	1	0909	1	1748.60	7.46595e+01	1897	1646	ta01	1	0909	1	2383.30	8.60628e+01	2508	2235
ft10	1	0909	2	1807.80	6.63503e+01	1917	1720	ta01	1	0909	2	2479.70	6.67938e+01	2582	2358
ft10	2	0909	1	1692.60	4.10371e+01	1754	1627	ta01	2	0909	1	2389.20	7.43812e+01	2535	2278
ft10	2	0909	2	1796.30	3.84943e+01	1861	1748	ta01	2	0909	2	2452.10	5.28951e+01	2532	2367
ft10	6	0909	1	1740.90	6.06769e+01	1866	1662	ta01	6	0909	1	2316.70	8.19842e+01	2492	2177
ft10	6	0909	2	1845.00	4.92118e+01	1926	1756	ta01	6	0909	2	2519.70	6.82701e+01	2632	2391
ft20	1	0606	1	2167.80	6.70981e+01	2313	2079	abz7	1	0606	1	1382.60	4.46256e+01	1475	1327
ft20	1	0606	2	2271.50	7.59924e+01	2415	2148	abz7	1	0606	2	1382.70	2.73571e+01	1434	1346
ft20	2	0606	1	2180.60	9.41650e+01	2296	2016	abz7	2	0606	1	1380.60	2.65977e+01	1407	1327
ft20	2	0606	2	2243.00	7.92856e+01	2387	2106	abz7	2	0606	2	1395.20	2.73964e+01	1450	1365
ft20	6	0606	1	2192.70	5.15501e+01	2268	2115	abz7	6	0606	1	1370.80	3.18773e+01	1431	1313
ft20	6	0606	2	2233.40	8.76153e+01	2363	2094	abz7	6	0606	2	1394.90	1.77845e+01	1426	1370
ft20	1	0609	1	2051.60	1.21272e+02	2203	1907	abz7	1	0609	1	1333.90	3.04514e+01	1400	1297
ft20	1	0609	2	2251.40	8.76712e+01	2382	2115	abz7	1	0609	2	1400.20	4.34415e+01	1501	1345
ft20	2	0609	1	2070.80	9.65524e+01	2229	1933	abz7	2	0609	1	1350.60	4.68000e+01	1453	1272
ft20	2	0609	2	2252.10	9.82491e+01	2493	2160	abz7	2	0609	2	1400.20	3.40376e+01	1457	1339
ft20	6	0609	1	2065.10	7.59519e+01	2176	1928	abz7	6	0609	1	1322.40	3.72993e+01	1414	1283
ft20	6	0609	2	2264.70	9.17846e+01	2458	2121	abz7	6	0609	2	1385.00	2.35669e+01	1435	1348
ft20	1	0906	1	2147.20	8.53063e+01	2250	2000	abz7	1	0906	1	1336.20	3.98693e+01	1396	1263
ft20	1	0906	2	2246.70	5.27505e+01	2327	2141	abz7	1	0906	2	1391.40	1.38434e+01	1414	1367
ft20	2	0906	1	2150.90	7.22059e+01	2318	2055	abz7	2	0906	1	1361.50	3.72485e+01	1440	1321
ft20	2	0906	2	2319.20	7.92058e+01	2458	2210	abz7	2	0906	2	1403.70	3.10807e+01	1491	1375
ft20	6	0906	1	2146.10	8.61457e+01	2264	1968	abz7	6	0906	1	1324.90	2.22865e+01	1372	1304
ft20	6	0906	2	2278.20	8.83796e+01	2458	2163	abz7	6	0906	2	1394.70	3.26529e+01	1463	1339
ft20	1	0909	1	2047.90	1.19620e+02	2267	1903	abz7	1	0909	1	1288.70	4.98398e+01	1382	1208
ft20	1	0909	2	2189.80	9.54765e+01	2346	2023	abz7	1	0909	2	1376.00	3.44674e+01	1425	1316
ft20	2	0909	1	1967.40	1.03413e+02	2120	1843	abz7	2	0909	1	1311.00	7.12629e+01	1426	1185
ft20	2	0909	2	2201.80	8.38234e+01	2341	2077	abz7	2	0909	2	1379.20	2.22432e+01	1406	1340
ft20	6	0909	1	2011.30	7.91695e+01	2165	1919	abz7	6	0909	1	1300.40	5.76493e+01	1391	1219
ft20	6	0909	2	2204.10	6.44150e+01	2315	2098	abz7	6	0909	2	1360.40	1.77268e+01	1392	1336
inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min

A.3.2.3 Critère $\Pi'_{\mathbf{moy}}$

inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	1434.00	6.63325e+00	1440.00	1420.00
ft10	1	0606	2	1475.00	5.00000e+00	1480.00	1470.00
ft10	2	0606	1	1439.00	5.38516e+00	1450.00	1430.00
ft10	2	0606	2	1477.00	6.40312e+00	1490.00	1470.00
ft10	6	0606	1	1444.00	8.00000e+00	1460.00	1430.00
ft10	6	0606	2	1480.00	7.74597e+00	1490.00	1460.00
ft10	1	0609	1	1410.00	4.47214e+00	1420.00	1400.00
ft10	1	0609	2	1455.00	6.70820e+00	1470.00	1450.00
ft10	2	0609	1	1411.00	3.00000e+00	1420.00	1410.00
ft10	2	0609	2	1452.00	6.00000e+00	1460.00	1440.00
ft10	6	0609	1	1417.00	4.58258e+00	1420.00	1410.00
ft10	6	0609	2	1455.00	5.00000e+00	1460.00	1450.00
ft10	1	0906	1	1417.00	4.58258e+00	1420.00	1410.00
ft10	1	0906	2	1458.00	6.00000e+00	1470.00	1450.00
ft10	2	0906	1	1420.00	4.47214e+00	1430.00	1410.00
ft10	2	0906	2	1462.00	7.48331e+00	1470.00	1450.00
ft10	6	0906	1	1432.00	4.00000e+00	1440.00	1430.00
ft10	6	0906	2	1467.00	4.58258e+00	1470.00	1460.00
ft10	1	0909	1	1401.00	3.00000e+00	1410.00	1400.00
ft10	1	0909	2	1433.00	4.58258e+00	1440.00	1430.00
ft10	2	0909	1	1406.00	4.89898e+00	1410.00	1400.00
ft10	2	0909	2	1434.00	6.63325e+00	1440.00	1420.00
ft10	6	0909	1	1416.00	4.89898e+00	1420.00	1410.00
ft10	6	0909	2	1441.00	5.38516e+00	1450.00	1430.00
ft20	1	0606	1	1647.00	6.40312e+00	1660.00	1640.00
ft20	1	0606	2	1692.00	7.48331e+00	1700.00	1680.00
ft20	2	0606	1	1649.00	7.00000e+00	1660.00	1640.00
ft20	2	0606	2	1691.00	7.00000e+00	1700.00	1680.00
ft20	6	0606	1	1653.00	4.58258e+00	1660.00	1650.00
ft20	6	0606	2	1695.00	8.06226e+00	1710.00	1680.00
ft20	1	0609	1	1610.00	0.00000e+00	1610.00	1610.00
ft20	1	0609	2	1663.00	7.81025e+00	1680.00	1650.00
ft20	2	0609	1	1616.00	4.89898e+00	1620.00	1610.00
ft20	2	0609	2	1664.00	4.89898e+00	1670.00	1660.00
ft20	6	0609	1	1615.00	5.00000e+00	1620.00	1610.00
ft20	6	0609	2	1665.00	6.70820e+00	1680.00	1660.00
ft20	1	0906	1	1636.00	4.89898e+00	1640.00	1630.00
ft20	1	0906	2	1675.00	5.00000e+00	1680.00	1670.00
ft20	2	0906	1	1640.00	4.47214e+00	1650.00	1630.00
ft20	2	0906	2	1677.00	1.00499e+01	1700.00	1660.00
ft20	6	0906	1	1638.00	4.00000e+00	1640.00	1630.00
ft20	6	0906	2	1675.00	8.06226e+00	1690.00	1660.00
ft20	1	0909	1	1611.00	3.00000e+00	1620.00	1610.00
ft20	1	0909	2	1642.00	6.00000e+00	1650.00	1630.00
ft20	2	0909	1	1617.00	4.58258e+00	1620.00	1610.00
ft20	2	0909	2	1647.00	6.40312e+00	1660.00	1640.00
ft20	6	0909	1	1619.00	3.00000e+00	1620.00	1610.00
ft20	6	0909	2	1646.00	4.89898e+00	1650.00	1640.00

inst	op	taux	sch	moy	écart	max	min
ta01	1	0606	1	1804.00	1.35647e+01	1830.00	1780.00
ta01	1	0606	2	1931.00	1.37477e+01	1950.00	1910.00
ta01	2	0606	1	1796.00	1.35647e+01	1820.00	1770.00
ta01	2	0606	2	1931.00	1.22066e+01	1950.00	1910.00
ta01	6	0606	1	1817.00	1.18743e+01	1840.00	1800.00
ta01	6	0606	2	1940.00	1.34164e+01	1960.00	1910.00
ta01	1	0609	1	1686.00	6.63325e+00	1700.00	1680.00
ta01	1	0609	2	1859.00	1.30000e+01	1880.00	1830.00
ta01	2	0609	1	1694.00	1.11355e+01	1710.00	1680.00
ta01	2	0609	2	1857.00	6.40312e+00	1870.00	1850.00
ta01	6	0609	1	1710.00	1.00000e+01	1720.00	1690.00
ta01	6	0609	2	1863.00	9.00000e+00	1880.00	1850.00
ta01	1	0906	1	1764.00	6.63325e+00	1770.00	1750.00
ta01	1	0906	2	1888.00	9.79796e+00	1910.00	1870.00
ta01	2	0906	1	1776.00	6.63325e+00	1790.00	1770.00
ta01	2	0906	2	1901.00	1.30000e+01	1920.00	1870.00
ta01	6	0906	1	1803.00	1.00499e+01	1820.00	1780.00
ta01	6	0906	2	1902.00	9.79796e+00	1910.00	1880.00
ta01	1	0909	1	1688.00	4.00000e+00	1690.00	1680.00
ta01	1	0909	2	1811.00	7.00000e+00	1820.00	1800.00
ta01	2	0909	1	1692.00	4.00000e+00	1700.00	1690.00
ta01	2	0909	2	1817.00	6.40312e+00	1830.00	1810.00
ta01	6	0909	1	1723.00	6.40312e+00	1740.00	1720.00
ta01	6	0909	2	1825.00	9.21954e+00	1840.00	1810.00
abz7	1	0606	1	970.60	6.66633e+00	984.00	960.00
abz7	1	0606	2	1052.00	9.79796e+00	1070.00	1040.00
abz7	2	0606	1	975.20	5.43691e+00	986.00	965.00
abz7	2	0606	2	1061.00	8.30662e+00	1070.00	1050.00
abz7	6	0606	1	980.60	7.87655e+00	998.00	972.00
abz7	6	0606	2	1056.00	8.00000e+00	1070.00	1040.00
abz7	1	0609	1	911.60	3.03974e+00	916.00	908.00
abz7	1	0609	2	1002.70	6.55820e+00	1020.00	999.00
abz7	2	0609	1	912.10	5.41202e+00	921.00	904.00
abz7	2	0609	2	1013.00	4.58258e+00	1020.00	1010.00
abz7	6	0609	1	917.80	3.62767e+00	924.00	910.00
abz7	6	0609	2	1006.00	4.89898e+00	1010.00	1000.00
abz7	1	0906	1	947.70	7.44379e+00	961.00	934.00
abz7	1	0906	2	1023.00	1.10000e+01	1040.00	1000.00
abz7	2	0906	1	951.90	5.10784e+00	963.00	944.00
abz7	2	0906	2	1033.00	6.40312e+00	1040.00	1020.00
abz7	6	0906	1	960.10	6.51844e+00	971.00	947.00
abz7	6	0906	2	1030.00	8.94427e+00	1040.00	1010.00
abz7	1	0909	1	905.60	3.80000e+00	913.00	899.00
abz7	1	0909	2	972.60	5.64269e+00	983.00	963.00
abz7	2	0909	1	907.70	4.05093e+00	914.00	899.00
abz7	2	0909	2	984.00	3.43511e+00	989.00	977.00
abz7	6	0909	1	918.70	3.90000e+00	926.00	914.00
abz7	6	0909	2	980.50	3.61248e+00	985.00	973.00

A.3.2.4 Critère $\Pi'_{\text{écart}}$

inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	91.39	8.01641e+00	106.000	76.900
ft10	1	0606	2	113.80	6.11228e+00	122.000	104.000
ft10	2	0606	1	92.21	4.22006e+00	101.000	86.100
ft10	2	0606	2	111.70	3.16386e+00	116.000	107.000
ft10	6	0606	1	93.36	6.33107e+00	105.000	83.700
ft10	6	0606	2	112.60	6.91665e+00	124.000	101.000
ft10	1	0609	1	59.13	6.48507e+00	67.900	49.100
ft10	1	0609	2	101.48	6.80849e+00	110.000	87.100
ft10	2	0609	1	53.88	3.87396e+00	59.800	45.300
ft10	2	0609	2	100.28	4.44428e+00	107.000	93.800
ft10	6	0609	1	64.75	4.21503e+00	70.200	58.500
ft10	6	0609	2	100.12	3.17106e+00	105.000	94.500
ft10	1	0906	1	82.18	5.87449e+00	90.100	71.200
ft10	1	0906	2	106.90	3.11288e+00	113.000	102.000
ft10	2	0906	1	79.17	3.58191e+00	83.600	72.500
ft10	2	0906	2	106.82	4.85506e+00	114.000	97.200
ft10	6	0906	1	87.97	1.80834e+00	91.300	84.900
ft10	6	0906	2	104.50	5.25833e+00	112.000	93.000
ft10	1	0909	1	59.27	3.60335e+00	64.600	54.400
ft10	1	0909	2	89.23	5.00760e+00	97.000	79.700
ft10	2	0909	1	51.04	4.10760e+00	57.900	46.200
ft10	2	0909	2	91.57	3.85514e+00	96.800	84.600
ft10	6	0909	1	66.19	1.52345e+00	68.900	64.300
ft10	6	0909	2	90.88	4.83566e+00	98.700	82.700
ft20	1	0606	1	107.34	5.36287e+00	117.000	99.400
ft20	1	0606	2	136.40	8.10185e+00	146.000	117.000
ft20	2	0606	1	109.07	6.35327e+00	121.000	97.700
ft20	2	0606	2	138.60	6.48383e+00	149.000	130.000
ft20	6	0606	1	111.30	4.77598e+00	118.000	104.000
ft20	6	0606	2	137.30	1.05835e+01	152.000	119.000
ft20	1	0609	1	62.92	7.75368e+00	74.500	52.900
ft20	1	0609	2	125.60	4.88262e+00	134.000	117.000
ft20	2	0609	1	68.24	8.31038e+00	86.700	57.300
ft20	2	0609	2	125.00	6.55744e+00	140.000	117.000
ft20	6	0609	1	69.41	9.73288e+00	84.400	54.300
ft20	6	0609	2	129.90	7.10563e+00	142.000	114.000
ft20	1	0906	1	92.20	4.26802e+00	96.600	82.300
ft20	1	0906	2	129.60	7.44580e+00	140.000	119.000
ft20	2	0906	1	94.15	5.94664e+00	104.000	86.700
ft20	2	0906	2	132.40	6.49923e+00	140.000	120.000
ft20	6	0906	1	91.67	4.55589e+00	98.300	84.200
ft20	6	0906	2	132.80	1.02840e+01	152.000	115.000
ft20	1	0909	1	58.48	6.02342e+00	70.400	49.600
ft20	1	0909	2	110.80	6.19355e+00	121.000	101.000
ft20	2	0909	1	57.23	4.51930e+00	62.900	50.200
ft20	2	0909	2	111.69	6.97602e+00	125.000	98.900
ft20	6	0909	1	63.56	7.00131e+00	73.000	51.600
ft20	6	0909	2	112.10	5.26213e+00	119.000	102.000

inst	op	taux	sch	moy	écart	max	min
ta01	1	0606	1	220.60	1.06508e+01	242.000	209.000
ta01	1	0606	2	248.60	8.11419e+00	262.000	238.000
ta01	2	0606	1	217.50	9.12414e+00	237.000	204.000
ta01	2	0606	2	248.80	9.58958e+00	271.000	237.000
ta01	6	0606	1	216.20	8.20731e+00	231.000	201.000
ta01	6	0606	2	251.20	5.01597e+00	260.000	243.000
ta01	1	0609	1	122.60	1.40014e+01	152.000	103.000
ta01	1	0609	2	240.60	6.71118e+00	251.000	226.000
ta01	2	0609	1	135.10	1.48893e+01	158.000	115.000
ta01	2	0609	2	243.80	6.06300e+00	255.000	236.000
ta01	6	0609	1	132.30	1.21659e+01	145.000	104.000
ta01	6	0609	2	244.50	8.77781e+00	264.000	232.000
ta01	1	0906	1	173.00	8.20975e+00	185.000	159.000
ta01	1	0906	2	229.70	5.58659e+00	240.000	221.000
ta01	2	0906	1	182.70	5.91692e+00	194.000	173.000
ta01	2	0906	2	230.90	8.20305e+00	243.000	214.000
ta01	6	0906	1	178.60	5.91946e+00	188.000	168.000
ta01	6	0906	2	229.90	6.92026e+00	239.000	215.000
ta01	1	0909	1	111.99	7.67990e+00	121.000	92.900
ta01	1	0909	2	202.80	5.82752e+00	217.000	194.000
ta01	2	0909	1	115.00	6.52687e+00	128.000	107.000
ta01	2	0909	2	210.20	6.02993e+00	222.000	200.000
ta01	6	0909	1	126.80	6.46220e+00	140.000	118.000
ta01	6	0909	2	209.10	8.09259e+00	225.000	200.000
abz7	1	0606	1	128.90	6.00750e+00	139.000	117.000
abz7	1	0606	2	154.00	3.13050e+00	158.000	149.000
abz7	2	0606	1	133.60	2.87054e+00	141.000	130.000
abz7	2	0606	2	156.00	1.84391e+00	159.000	152.000
abz7	6	0606	1	129.40	4.69468e+00	137.000	120.000
abz7	6	0606	2	155.20	2.35797e+00	160.000	152.000
abz7	1	0609	1	75.59	5.47256e+00	82.500	65.600
abz7	1	0609	2	146.90	2.42693e+00	153.000	144.000
abz7	2	0609	1	78.96	9.80583e+00	93.500	63.900
abz7	2	0609	2	149.40	3.07246e+00	153.000	144.000
abz7	6	0609	1	74.43	6.64365e+00	83.400	60.600
abz7	6	0609	2	145.50	4.10488e+00	151.000	138.000
abz7	1	0906	1	99.58	5.93579e+00	111.000	87.200
abz7	1	0906	2	142.60	3.66606e+00	146.000	134.000
abz7	2	0906	1	107.50	4.63141e+00	116.000	100.000
abz7	2	0906	2	145.50	3.04138e+00	150.000	141.000
abz7	6	0906	1	99.64	6.13208e+00	111.000	89.100
abz7	6	0906	2	141.50	3.44238e+00	146.000	133.000
abz7	1	0909	1	58.06	5.89766e+00	69.200	46.000
abz7	1	0909	2	118.90	5.90678e+00	130.000	110.000
abz7	2	0909	1	64.68	8.07265e+00	77.400	47.300
abz7	2	0909	2	128.40	2.28910e+00	133.000	125.000
abz7	6	0909	1	65.48	6.69713e+00	76.800	56.000
abz7	6	0909	2	122.60	3.63868e+00	129.000	116.000

A.3.3 Résultats pour 500 individus après 20 générations

A.3.3.1 Critère $\Pi'_{\min}$

inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	1094.30	1.25861e+01	1110	1070	ta01	1	0606	1	1444.40	1.02489e+01	1460	1428
ft10	1	0606	2	1078.20	1.99940e+01	1104	1040	ta01	1	0606	2	1429.10	1.96390e+01	1453	1394
ft10	2	0606	1	1091.40	1.18507e+01	1111	1075	ta01	2	0606	1	1449.40	7.77432e+00	1463	1436
ft10	2	0606	2	1080.50	1.71595e+01	1119	1056	ta01	2	0606	2	1425.90	1.34048e+01	1446	1404
ft10	6	0606	1	1098.30	1.41425e+01	1122	1078	ta01	6	0606	1	1450.80	8.94204e+00	1466	1439
ft10	6	0606	2	1086.60	1.72058e+01	1111	1051	ta01	6	0606	2	1443.90	6.37887e+00	1454	1434
ft10	1	0609	1	1123.70	1.04312e+01	1140	1109	ta01	1	0609	1	1461.40	1.18169e+01	1481	1436
ft10	1	0609	2	1084.40	1.75795e+01	1111	1057	ta01	1	0609	2	1436.00	1.12161e+01	1452	1417
ft10	2	0609	1	1108.90	1.48960e+01	1129	1085	ta01	2	0609	1	1457.40	1.07629e+01	1475	1438
ft10	2	0609	2	1075.70	1.27675e+01	1087	1042	ta01	2	0609	2	1432.70	1.00105e+01	1449	1417
ft10	6	0609	1	1121.90	9.84327e+00	1143	1106	ta01	6	0609	1	1457.40	2.18596e+01	1491	1402
ft10	6	0609	2	1092.10	9.01610e+00	1100	1071	ta01	6	0609	2	1450.10	9.84327e+00	1463	1431
ft10	1	0906	1	1088.40	1.52656e+01	1106	1059	ta01	1	0906	1	1440.60	8.82270e+00	1454	1425
ft10	1	0906	2	1076.70	9.56086e+00	1096	1062	ta01	1	0906	2	1423.80	1.34075e+01	1443	1404
ft10	2	0906	1	1083.40	1.16207e+01	1098	1064	ta01	2	0906	1	1442.00	8.79773e+00	1455	1427
ft10	2	0906	2	1062.30	1.17818e+01	1082	1046	ta01	2	0906	2	1424.50	8.39345e+00	1438	1411
ft10	6	0906	1	1095.20	1.44000e+01	1113	1059	ta01	6	0906	1	1442.40	8.87919e+00	1451	1420
ft10	6	0906	2	1079.00	8.12404e+00	1091	1060	ta01	6	0906	2	1436.60	1.92624e+01	1469	1394
ft10	1	0909	1	1110.70	1.31229e+01	1128	1085	ta01	1	0909	1	1458.20	1.04766e+01	1479	1441
ft10	1	0909	2	1078.30	8.83233e+00	1098	1066	ta01	1	0909	2	1425.70	1.21906e+01	1443	1404
ft10	2	0909	1	1109.30	1.49335e+01	1133	1079	ta01	2	0909	1	1458.70	1.06306e+01	1477	1442
ft10	2	0909	2	1079.00	1.08904e+01	1097	1057	ta01	2	0909	2	1431.90	9.10439e+00	1448	1421
ft10	6	0909	1	1131.70	1.01789e+01	1144	1113	ta01	6	0909	1	1462.80	1.14088e+01	1478	1443
ft10	6	0909	2	1092.40	1.32680e+01	1119	1073	ta01	6	0909	2	1432.70	2.06158e+01	1463	1398
ft20	1	0606	1	1357.30	2.26586e+01	1385	1323	abz7	1	0606	1	779.30	5.08035e+00	787	771
ft20	1	0606	2	1338.40	1.43889e+01	1370	1320	abz7	1	0606	2	781.60	2.76405e+00	787	778
ft20	2	0606	1	1365.10	2.20837e+01	1403	1329	abz7	2	0606	1	782.10	3.85876e+00	786	772
ft20	2	0606	2	1323.50	1.63966e+01	1351	1288	abz7	2	0606	2	774.90	4.59239e+00	780	766
ft20	6	0606	1	1356.90	2.44518e+01	1384	1304	abz7	6	0606	1	786.00	5.69210e+00	793	777
ft20	6	0606	2	1336.20	1.93432e+01	1375	1302	abz7	6	0606	2	780.60	9.47840e+00	791	758
ft20	1	0609	1	1405.20	2.13626e+01	1425	1362	abz7	1	0609	1	788.90	6.13922e+00	797	776
ft20	1	0609	2	1342.30	1.86389e+01	1369	1311	abz7	1	0609	2	779.60	4.43170e+00	785	772
ft20	2	0609	1	1406.30	2.11142e+01	1430	1376	abz7	2	0609	1	787.90	3.96106e+00	793	782
ft20	2	0609	2	1338.70	2.63782e+01	1386	1288	abz7	2	0609	2	776.70	6.72384e+00	784	761
ft20	6	0609	1	1411.90	1.35532e+01	1432	1385	abz7	6	0609	1	790.70	6.95773e+00	800	775
ft20	6	0609	2	1339.70	1.98497e+01	1372	1305	abz7	6	0609	2	784.00	3.37639e+00	789	778
ft20	1	0906	1	1353.50	1.76932e+01	1377	1314	abz7	1	0906	1	783.60	3.95474e+00	791	777
ft20	1	0906	2	1321.70	2.05331e+01	1359	1292	abz7	1	0906	2	773.90	6.48768e+00	781	763
ft20	2	0906	1	1357.30	1.36459e+01	1382	1325	abz7	2	0906	1	782.10	4.43734e+00	792	775
ft20	2	0906	2	1322.60	1.91635e+01	1351	1288	abz7	2	0906	2	774.60	7.18610e+00	784	758
ft20	6	0906	1	1360.80	1.22049e+01	1379	1344	abz7	6	0906	1	791.50	4.22493e+00	800	786
ft20	6	0906	2	1339.00	1.93442e+01	1370	1313	abz7	6	0906	2	778.70	9.11098e+00	792	766
ft20	1	0909	1	1402.30	1.30694e+01	1416	1379	abz7	1	0909	1	790.20	4.46766e+00	800	782
ft20	1	0909	2	1324.80	2.76651e+01	1360	1269	abz7	1	0909	2	777.10	4.78435e+00	783	766
ft20	2	0909	1	1399.00	2.50280e+01	1429	1343	abz7	2	0909	1	787.80	4.06940e+00	794	781
ft20	2	0909	2	1312.80	2.18486e+01	1338	1272	abz7	2	0909	2	777.30	6.00083e+00	788	768
ft20	6	0909	1	1407.00	1.47105e+01	1428	1374	abz7	6	0909	1	790.20	8.95321e+00	801	777
ft20	6	0909	2	1328.70	1.95809e+01	1370	1308	abz7	6	0909	2	781.60	6.10246e+00	790	769
inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min

A.3.3.2 Critère $\Pi'_{\max}$

inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	1259.20	1.29985e+01	1286	1240	ta01	1	0606	1	1578.30	9.96042e+00	1595	1566
ft10	1	0606	2	1169.20	8.60000e+00	1180	1152	ta01	1	0606	2	1504.70	9.83921e+00	1518	1481
ft10	2	0606	1	1247.00	9.31665e+00	1262	1228	ta01	2	0606	1	1579.50	7.36546e+00	1598	1572
ft10	2	0606	2	1156.70	1.30081e+01	1181	1139	ta01	2	0606	2	1506.20	9.57914e+00	1525	1492
ft10	6	0606	1	1278.60	1.71942e+01	1316	1252	ta01	6	0606	1	1599.90	8.61916e+00	1618	1583
ft10	6	0606	2	1179.90	1.09586e+01	1195	1160	ta01	6	0606	2	1525.80	8.86341e+00	1543	1512
ft10	1	0609	1	1303.10	1.09403e+01	1320	1284	ta01	1	0609	1	1607.40	8.66256e+00	1620	1593
ft10	1	0609	2	1206.10	1.21116e+01	1238	1196	ta01	1	0609	2	1536.70	8.69540e+00	1549	1522
ft10	2	0609	1	1294.10	7.21734e+00	1306	1282	ta01	2	0609	1	1604.60	6.20000e+00	1617	1596
ft10	2	0609	2	1201.50	1.68538e+01	1236	1178	ta01	2	0609	2	1540.60	6.35925e+00	1549	1530
ft10	6	0609	1	1310.60	5.49909e+00	1317	1298	ta01	6	0609	1	1624.80	7.05408e+00	1640	1616
ft10	6	0609	2	1216.50	8.45281e+00	1229	1206	ta01	6	0609	2	1559.60	7.80000e+00	1569	1544
ft10	1	0906	1	1270.80	1.20648e+01	1284	1248	ta01	1	0906	1	1583.00	9.45516e+00	1599	1569
ft10	1	0906	2	1169.50	1.46646e+01	1203	1153	ta01	1	0906	2	1506.50	1.06325e+01	1526	1484
ft10	2	0906	1	1252.90	1.19453e+01	1274	1239	ta01	2	0906	1	1576.30	6.79779e+00	1590	1566
ft10	2	0906	2	1163.60	1.11463e+01	1193	1152	ta01	2	0906	2	1508.60	8.86792e+00	1528	1492
ft10	6	0906	1	1285.10	8.20305e+00	1299	1267	ta01	6	0906	1	1616.90	9.75141e+00	1632	1599
ft10	6	0906	2	1192.00	1.07051e+01	1212	1178	ta01	6	0906	2	1530.60	1.89747e+01	1575	1516
ft10	1	0909	1	1305.80	1.17796e+01	1326	1292	ta01	1	0909	1	1607.10	7.91770e+00	1623	1600
ft10	1	0909	2	1214.90	1.20453e+01	1248	1206	ta01	1	0909	2	1558.10	1.31488e+01	1591	1541
ft10	2	0909	1	1296.70	1.21413e+01	1324	1282	ta01	2	0909	1	1608.30	8.07527e+00	1625	1598
ft10	2	0909	2	1207.10	1.48017e+01	1243	1188	ta01	2	0909	2	1548.90	5.78705e+00	1563	1540
ft10	6	0909	1	1317.70	1.16022e+01	1348	1303	ta01	6	0909	1	1631.60	7.93977e+00	1644	1621
ft10	6	0909	2	1235.10	7.34098e+00	1250	1226	ta01	6	0909	2	1571.30	9.55039e+00	1590	1558
ft20	1	0606	1	1523.50	8.51176e+00	1545	1513	abz7	1	0606	1	840.10	3.85876e+00	847	833
ft20	1	0606	2	1417.40	1.85213e+01	1439	1383	abz7	1	0606	2	810.80	2.44131e+00	814	805
ft20	2	0606	1	1532.60	8.58138e+00	1545	1515	abz7	2	0606	1	840.50	2.33452e+00	845	837
ft20	2	0606	2	1421.40	1.43052e+01	1452	1407	abz7	2	0606	2	812.20	3.73631e+00	818	805
ft20	6	0606	1	1530.10	1.18865e+01	1552	1515	abz7	6	0606	1	847.10	3.93573e+00	854	840
ft20	6	0606	2	1423.70	1.45193e+01	1452	1402	abz7	6	0606	2	815.00	5.88218e+00	828	807
ft20	1	0609	1	1568.70	7.43034e+00	1584	1559	abz7	1	0609	1	851.00	2.52982e+00	856	848
ft20	1	0609	2	1474.80	9.42125e+00	1484	1452	abz7	1	0609	2	824.90	3.67287e+00	832	819
ft20	2	0609	1	1568.00	3.49285e+00	1573	1563	abz7	2	0609	1	851.00	4.17133e+00	861	846
ft20	2	0609	2	1473.80	9.66230e+00	1491	1463	abz7	2	0609	2	824.00	3.40588e+00	829	819
ft20	6	0609	1	1569.40	4.36348e+00	1575	1561	abz7	6	0609	1	853.70	2.64764e+00	857	850
ft20	6	0609	2	1480.60	1.23790e+01	1500	1462	abz7	6	0609	2	829.70	3.84838e+00	836	824
ft20	1	0906	1	1528.80	5.84466e+00	1537	1517	abz7	1	0906	1	843.60	3.00666e+00	850	839
ft20	1	0906	2	1429.40	1.76136e+01	1456	1387	abz7	1	0906	2	813.40	3.26190e+00	817	807
ft20	2	0906	1	1530.10	7.55579e+00	1543	1519	abz7	2	0906	1	843.40	2.87054e+00	848	838
ft20	2	0906	2	1432.40	2.06843e+01	1459	1396	abz7	2	0906	2	814.60	3.69324e+00	820	808
ft20	6	0906	1	1537.80	1.23434e+01	1557	1521	abz7	6	0906	1	854.80	1.98997e+00	860	853
ft20	6	0906	2	1443.20	1.42253e+01	1466	1415	abz7	6	0906	2	818.30	4.73392e+00	828	809
ft20	1	0909	1	1567.70	5.69298e+00	1577	1557	abz7	1	0909	1	852.40	2.00998e+00	855	850
ft20	1	0909	2	1474.00	1.26965e+01	1489	1445	abz7	1	0909	2	828.70	1.79165e+00	832	825
ft20	2	0909	1	1571.10	6.13922e+00	1585	1562	abz7	2	0909	1	852.50	2.41868e+00	856	848
ft20	2	0909	2	1471.20	1.53610e+01	1494	1442	abz7	2	0909	2	827.50	2.53969e+00	830	823
ft20	6	0909	1	1572.90	4.06079e+00	1579	1564	abz7	6	0909	1	858.30	2.53180e+00	864	855
ft20	6	0909	2	1484.40	1.06790e+01	1509	1471	abz7	6	0909	2	836.70	4.22019e+00	845	831
inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min

A.3.3.3 Critère $\Pi'_{\mathbf{moy}}$

inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	1166.00	4.89898e+00	1170.00	1160.00
ft10	1	0606	2	1126.00	8.00000e+00	1140.00	1120.00
ft10	2	0606	1	1161.00	5.38516e+00	1170.00	1150.00
ft10	2	0606	2	1117.00	1.18743e+01	1140.00	1100.00
ft10	6	0606	1	1183.00	6.40312e+00	1190.00	1170.00
ft10	6	0606	2	1129.00	9.43398e+00	1140.00	1110.00
ft10	1	0609	1	1210.00	0.00000e+00	1210.00	1210.00
ft10	1	0609	2	1136.00	1.11355e+01	1150.00	1110.00
ft10	2	0609	1	1209.00	3.00000e+00	1210.00	1200.00
ft10	2	0609	2	1129.00	1.22066e+01	1140.00	1100.00
ft10	6	0609	1	1220.00	0.00000e+00	1220.00	1220.00
ft10	6	0609	2	1147.00	4.58258e+00	1150.00	1140.00
ft10	1	0906	1	1169.00	5.38516e+00	1180.00	1160.00
ft10	1	0906	2	1118.00	1.24900e+01	1130.00	1090.00
ft10	2	0906	1	1164.00	6.63325e+00	1170.00	1150.00
ft10	2	0906	2	1108.00	1.07703e+01	1120.00	1090.00
ft10	6	0906	1	1192.00	4.00000e+00	1200.00	1190.00
ft10	6	0906	2	1123.00	1.00499e+01	1140.00	1110.00
ft10	1	0909	1	1217.00	4.58258e+00	1220.00	1210.00
ft10	1	0909	2	1145.00	5.00000e+00	1150.00	1140.00
ft10	2	0909	1	1210.00	0.00000e+00	1210.00	1210.00
ft10	2	0909	2	1137.00	6.40312e+00	1150.00	1130.00
ft10	6	0909	1	1230.00	0.00000e+00	1230.00	1230.00
ft10	6	0909	2	1161.00	5.38516e+00	1170.00	1150.00
ft20	1	0606	1	1437.00	6.40312e+00	1450.00	1430.00
ft20	1	0606	2	1377.00	1.67631e+01	1400.00	1350.00
ft20	2	0606	1	1437.00	1.79165e+01	1460.00	1400.00
ft20	2	0606	2	1368.00	2.08806e+01	1390.00	1320.00
ft20	6	0606	1	1433.00	1.48661e+01	1450.00	1410.00
ft20	6	0606	2	1381.00	1.57797e+01	1410.00	1360.00
ft20	1	0609	1	1500.00	0.00000e+00	1500.00	1500.00
ft20	1	0609	2	1402.00	1.93907e+01	1430.00	1370.00
ft20	2	0609	1	1500.00	0.00000e+00	1500.00	1500.00
ft20	2	0609	2	1396.00	1.28062e+01	1410.00	1370.00
ft20	6	0609	1	1500.00	0.00000e+00	1500.00	1500.00
ft20	6	0609	2	1405.00	1.36015e+01	1430.00	1380.00
ft20	1	0906	1	1440.00	1.00000e+01	1460.00	1430.00
ft20	1	0906	2	1374.00	1.56205e+01	1390.00	1340.00
ft20	2	0906	1	1439.00	9.43398e+00	1450.00	1420.00
ft20	2	0906	2	1362.00	1.60000e+01	1390.00	1340.00
ft20	6	0906	1	1444.00	8.00000e+00	1460.00	1430.00
ft20	6	0906	2	1391.00	1.70000e+01	1410.00	1360.00
ft20	1	0909	1	1500.00	0.00000e+00	1500.00	1500.00
ft20	1	0909	2	1396.00	1.42829e+01	1410.00	1370.00
ft20	2	0909	1	1500.00	0.00000e+00	1500.00	1500.00
ft20	2	0909	2	1389.00	1.64012e+01	1410.00	1360.00
ft20	6	0909	1	1502.00	4.00000e+00	1510.00	1500.00
ft20	6	0909	2	1397.00	1.18743e+01	1420.00	1380.00

inst	op	taux	sch	moy	écart	max	min
ta01	1	0606	1	1506.00	4.89898e+00	1510.00	1500.00
ta01	1	0606	2	1464.00	1.49666e+01	1480.00	1440.00
ta01	2	0606	1	1508.00	4.00000e+00	1510.00	1500.00
ta01	2	0606	2	1465.00	6.70820e+00	1470.00	1450.00
ta01	6	0606	1	1522.00	4.00000e+00	1530.00	1520.00
ta01	6	0606	2	1476.00	6.63325e+00	1490.00	1470.00
ta01	1	0609	1	1540.00	0.00000e+00	1540.00	1540.00
ta01	1	0609	2	1483.00	6.40312e+00	1490.00	1470.00
ta01	2	0609	1	1540.00	0.00000e+00	1540.00	1540.00
ta01	2	0609	2	1479.00	5.38516e+00	1490.00	1470.00
ta01	6	0609	1	1550.00	0.00000e+00	1550.00	1550.00
ta01	6	0609	2	1492.00	8.71780e+00	1500.00	1480.00
ta01	1	0906	1	1510.00	0.00000e+00	1510.00	1510.00
ta01	1	0906	2	1463.00	6.40312e+00	1470.00	1450.00
ta01	2	0906	1	1509.00	3.00000e+00	1510.00	1500.00
ta01	2	0906	2	1462.00	8.71780e+00	1470.00	1440.00
ta01	6	0906	1	1536.00	4.89898e+00	1540.00	1530.00
ta01	6	0906	2	1477.00	9.00000e+00	1490.00	1460.00
ta01	1	0909	1	1540.00	0.00000e+00	1540.00	1540.00
ta01	1	0909	2	1482.00	6.00000e+00	1490.00	1470.00
ta01	2	0909	1	1540.00	0.00000e+00	1540.00	1540.00
ta01	2	0909	2	1488.00	4.00000e+00	1490.00	1480.00
ta01	6	0909	1	1559.00	3.00000e+00	1560.00	1550.00
ta01	6	0909	2	1503.00	4.58258e+00	1510.00	1500.00
abz7	1	0606	1	811.00	2.04939e+00	814.00	807.00
abz7	1	0606	2	794.90	2.42693e+00	799.00	790.00
abz7	2	0606	1	810.30	2.19317e+00	814.00	806.00
abz7	2	0606	2	791.70	2.10000e+00	795.00	788.00
abz7	6	0606	1	816.00	1.84391e+00	819.00	813.00
abz7	6	0606	2	797.20	2.18174e+00	801.00	793.00
abz7	1	0609	1	824.60	6.63325e-01	825.00	823.00
abz7	1	0609	2	800.20	2.27156e+00	803.00	796.00
abz7	2	0609	1	824.10	7.00000e-01	825.00	823.00
abz7	2	0609	2	799.00	3.54965e+00	803.00	791.00
abz7	6	0609	1	827.90	5.38516e-01	829.00	827.00
abz7	6	0609	2	804.00	1.78885e+00	807.00	801.00
abz7	1	0906	1	814.90	1.70000e+00	817.00	812.00
abz7	1	0906	2	792.20	3.99500e+00	797.00	784.00
abz7	2	0906	1	814.60	1.20000e+00	817.00	813.00
abz7	2	0906	2	793.50	3.26343e+00	796.00	787.00
abz7	6	0906	1	823.40	1.74356e+00	826.00	821.00
abz7	6	0906	2	796.90	3.26956e+00	802.00	792.00
abz7	1	0909	1	825.80	8.71780e-01	827.00	824.00
abz7	1	0909	2	802.60	1.49666e+00	804.00	799.00
abz7	2	0909	1	825.80	4.00000e-01	826.00	825.00
abz7	2	0909	2	802.10	2.21133e+00	806.00	797.00
abz7	6	0909	1	831.30	6.40312e-01	832.00	830.00
abz7	6	0909	2	809.50	1.36015e+00	813.00	808.00

A.3.3.4 Critère $\Pi'_{\text{écart}}$

inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	24.77	2.49842e+00	28.200	20.600
ft10	1	0606	2	15.09	6.78460e+00	24.400	7.220
ft10	2	0606	1	24.18	2.28771e+00	27.800	20.900
ft10	2	0606	2	10.68	3.02934e+00	15.100	4.440
ft10	6	0606	1	28.65	1.85970e+00	31.600	25.100
ft10	6	0606	2	13.00	3.47634e+00	18.500	7.360
ft10	1	0609	1	30.48	1.64791e+00	33.100	28.100
ft10	1	0609	2	18.76	2.42289e+00	22.900	14.600
ft10	2	0609	1	30.70	1.30461e+00	33.200	29.000
ft10	2	0609	2	21.01	5.08320e+00	29.000	14.300
ft10	6	0609	1	31.58	1.74000e+00	34.000	28.500
ft10	6	0609	2	20.12	1.58858e+00	22.700	16.700
ft10	1	0906	1	28.92	2.50472e+00	34.800	26.300
ft10	1	0906	2	14.96	2.49728e+00	19.900	11.000
ft10	2	0906	1	28.46	1.77831e+00	31.300	25.500
ft10	2	0906	2	15.69	4.68388e+00	25.000	9.610
ft10	6	0906	1	29.64	2.28175e+00	35.500	27.000
ft10	6	0906	2	17.99	5.19046e+00	30.400	11.800
ft10	1	0909	1	31.24	1.13243e+00	32.600	28.800
ft10	1	0909	2	21.35	1.83861e+00	23.400	17.900
ft10	2	0909	1	31.39	1.73173e+00	35.300	29.000
ft10	2	0909	2	21.00	2.74736e+00	26.000	16.500
ft10	6	0909	1	32.53	1.02572e+00	34.100	31.000
ft10	6	0909	2	22.87	2.70224e+00	28.200	19.200
ft20	1	0606	1	28.06	6.22370e+00	42.700	20.100
ft20	1	0606	2	11.84	3.50704e+00	17.800	4.060
ft20	2	0606	1	27.47	3.80475e+00	34.100	22.500
ft20	2	0606	2	15.77	4.00775e+00	23.900	10.200
ft20	6	0606	1	30.93	4.38134e+00	39.100	26.400
ft20	6	0606	2	13.18	1.81868e+00	17.200	10.800
ft20	1	0609	1	26.54	1.30476e+00	28.600	24.600
ft20	1	0609	2	19.90	4.79979e+00	29.400	13.900
ft20	2	0609	1	26.66	1.55576e+00	29.600	24.400
ft20	2	0609	2	22.94	3.59477e+00	28.900	18.200
ft20	6	0609	1	26.21	1.36268e+00	28.200	23.700
ft20	6	0609	2	23.18	4.92317e+00	30.500	12.400
ft20	1	0906	1	28.51	3.00880e+00	32.500	23.800
ft20	1	0906	2	16.25	3.76464e+00	22.700	10.700
ft20	2	0906	1	29.11	2.55165e+00	34.100	24.200
ft20	2	0906	2	14.68	3.44871e+00	20.600	9.160
ft20	6	0906	1	28.78	2.91369e+00	33.700	24.700
ft20	6	0906	2	14.82	2.10893e+00	18.200	11.500
ft20	1	0909	1	26.89	1.18191e+00	29.200	25.200
ft20	1	0909	2	22.68	3.27683e+00	28.200	16.000
ft20	2	0909	1	27.37	1.20171e+00	29.900	25.200
ft20	2	0909	2	24.71	4.68390e+00	32.700	18.200
ft20	6	0909	1	27.59	1.67419e+00	30.700	25.800
ft20	6	0909	2	26.43	4.74511e+00	33.000	16.800

inst	op	taux	sch	moy	écart	max	min
ta01	1	0606	1	21.84	2.19736e+00	25.700	18.300
ta01	1	0606	2	12.91	3.27132e+00	20.500	8.670
ta01	2	0606	1	21.83	2.35756e+00	26.800	18.900
ta01	2	0606	2	13.41	3.57036e+00	20.800	8.670
ta01	6	0606	1	25.22	1.96204e+00	29.300	22.300
ta01	6	0606	2	11.38	2.83582e+00	15.200	6.590
ta01	1	0609	1	24.63	1.68763e+00	26.900	21.800
ta01	1	0609	2	15.03	1.75787e+00	17.600	12.500
ta01	2	0609	1	25.35	9.13510e-01	26.600	23.900
ta01	2	0609	2	17.13	4.18761e+00	25.600	13.800
ta01	6	0609	1	26.81	1.14232e+00	28.400	24.900
ta01	6	0609	2	17.83	2.73022e+00	21.700	13.800
ta01	1	0906	1	23.64	1.67940e+00	27.500	21.500
ta01	1	0906	2	13.88	3.83348e+00	23.500	10.000
ta01	2	0906	1	23.61	1.50363e+00	27.700	21.700
ta01	2	0906	2	12.30	1.58051e+00	14.300	10.000
ta01	6	0906	1	26.78	1.30369e+00	28.200	24.000
ta01	6	0906	2	13.68	2.40200e+00	17.200	10.700
ta01	1	0909	1	25.24	1.20764e+00	27.400	23.800
ta01	1	0909	2	21.28	2.78632e+00	27.300	17.600
ta01	2	0909	1	24.88	5.52811e-01	25.900	24.000
ta01	2	0909	2	18.70	1.97737e+00	23.400	15.600
ta01	6	0909	1	27.47	1.13053e+00	29.400	26.000
ta01	6	0909	2	21.43	2.44215e+00	25.200	18.000
abz7	1	0606	1	10.22	1.84909e+00	14.400	7.870
abz7	1	0606	2	4.46	8.86402e-01	6.090	3.070
abz7	2	0606	1	9.97	1.26699e+00	13.100	8.610
abz7	2	0606	2	6.00	1.22380e+00	8.020	4.700
abz7	6	0606	1	10.35	1.07220e+00	12.000	8.800
abz7	6	0606	2	5.61	2.66779e+00	11.000	1.310
abz7	1	0609	1	10.05	6.93847e-01	11.300	9.130
abz7	1	0609	2	7.16	1.09740e+00	9.180	5.370
abz7	2	0609	1	10.39	5.89170e-01	11.300	9.480
abz7	2	0609	2	7.15	1.41780e+00	9.360	4.820
abz7	6	0609	1	10.11	4.02641e-01	10.500	9.210
abz7	6	0609	2	8.14	1.06069e+00	9.710	6.000
abz7	1	0906	1	10.03	7.71412e-01	11.500	9.000
abz7	1	0906	2	7.38	2.32633e+00	11.700	4.150
abz7	2	0906	1	10.06	6.44624e-01	11.200	8.880
abz7	2	0906	2	6.33	1.80209e+00	9.760	4.230
abz7	6	0906	1	10.64	8.25843e-01	12.100	9.480
abz7	6	0906	2	5.91	1.32789e+00	9.200	4.060
abz7	1	0909	1	10.01	5.56461e-01	11.200	9.340
abz7	1	0909	2	8.55	1.03481e+00	10.500	7.610
abz7	2	0909	1	10.08	3.40523e-01	10.900	9.740
abz7	2	0909	2	8.22	7.99806e-01	10.100	6.860
abz7	6	0909	1	10.76	5.40740e-01	11.500	10.000
abz7	6	0909	2	8.69	8.85813e-01	9.880	6.910

A.3.3.5 Critère $\Pi'_{\text{éval}}$

inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	9397.80	7.02379e+01	9491	9248
ft10	1	0606	2	10016.10	4.26015e+01	10080	9966
ft10	2	0606	1	15412.20	6.17200e+01	15493	15292
ft10	2	0606	2	13000.40	7.69873e+01	13138	12898
ft10	6	0606	1	9416.50	2.90457e+01	9466	9369
ft10	6	0606	2	9977.50	5.13308e+01	10113	9925
ft10	1	0609	1	10585.20	1.31057e+01	10607	10561
ft10	1	0609	2	12966.60	8.95647e+01	13079	12799
ft10	2	0609	1	16596.20	5.13766e+01	16670	16491
ft10	2	0609	2	15973.60	8.67816e+01	16128	15798
ft10	6	0609	1	10604.90	1.50960e+01	10628	10579
ft10	6	0609	2	13019.00	4.89285e+01	13081	12918
ft10	1	0906	1	10612.30	2.56517e+01	10664	10567
ft10	1	0906	2	11467.40	9.66356e+01	11621	11346
ft10	2	0906	1	19585.40	3.82262e+01	19659	19528
ft10	2	0906	2	15981.80	8.80929e+01	16128	15832
ft10	6	0906	1	10597.80	1.90095e+01	10623	10557
ft10	6	0906	2	11484.90	6.80786e+01	11598	11410
ft10	1	0909	1	10896.40	9.12360e+00	10913	10882
ft10	1	0909	2	14502.80	4.47008e+01	14585	14441
ft10	2	0909	1	19889.00	3.40969e+01	19934	19831
ft10	2	0909	2	19013.60	3.68217e+01	19082	18970
ft10	6	0909	1	10900.50	6.84471e+00	10913	10889
ft10	6	0909	2	14498.00	4.98137e+01	14591	14417
ft20	1	0606	1	9396.40	2.58232e+01	9435	9356
ft20	1	0606	2	10021.30	8.51928e+01	10178	9913
ft20	2	0606	1	15409.40	9.68857e+01	15607	15232
ft20	2	0606	2	12974.00	8.74666e+01	13128	12780
ft20	6	0606	1	9398.80	3.87448e+01	9466	9346
ft20	6	0606	2	10006.20	7.09377e+01	10078	9863
ft20	1	0609	1	10608.00	2.79571e+01	10656	10569
ft20	1	0609	2	12962.30	3.00468e+01	13013	12903
ft20	2	0609	1	16609.00	5.88914e+01	16706	16503
ft20	2	0609	2	16063.80	6.88619e+01	16204	15974
ft20	6	0609	1	10597.00	2.12415e+01	10623	10553
ft20	6	0609	2	12990.30	3.71835e+01	13042	12925
ft20	1	0906	1	10594.30	2.01646e+01	10635	10558
ft20	1	0906	2	11508.60	4.82145e+01	11569	11410
ft20	2	0906	1	19589.90	2.59054e+01	19646	19552
ft20	2	0906	2	15976.00	7.71699e+01	16114	15854
ft20	6	0906	1	10603.00	2.45031e+01	10630	10545
ft20	6	0906	2	11489.70	5.82753e+01	11566	11358
ft20	1	0909	1	10902.10	1.00643e+01	10914	10885
ft20	1	0909	2	14485.90	5.16845e+01	14577	14410
ft20	2	0909	1	19891.30	1.96573e+01	19912	19853
ft20	2	0909	2	19013.40	2.54252e+01	19054	18960
ft20	6	0909	1	10901.60	9.44669e+00	10918	10888
ft20	6	0909	2	14489.90	3.16495e+01	14531	14435

inst	op	taux	sch	moy	écart	max	min
ta01	1	0606	1	9385.60	3.25398e+01	9460	9348
ta01	1	0606	2	10018.50	7.98013e+01	10143	9835
ta01	2	0606	1	15409.30	1.02952e+02	15536	15193
ta01	2	0606	2	13038.20	1.01747e+02	13186	12868
ta01	6	0606	1	9394.10	3.65744e+01	9465	9318
ta01	6	0606	2	9997.20	7.05107e+01	10093	9855
ta01	1	0609	1	10595.40	2.29443e+01	10634	10555
ta01	1	0609	2	12977.40	7.27547e+01	13075	12819
ta01	2	0609	1	16592.10	6.11448e+01	16684	16477
ta01	2	0609	2	15989.00	6.78307e+01	16084	15870
ta01	6	0609	1	10605.70	2.23520e+01	10658	10580
ta01	6	0609	2	12964.00	6.29603e+01	13082	12886
ta01	1	0906	1	10596.90	2.73146e+01	10622	10533
ta01	1	0906	2	11501.20	5.39292e+01	11610	11422
ta01	2	0906	1	19582.70	4.04328e+01	19673	19526
ta01	2	0906	2	16001.00	8.10444e+01	16122	15886
ta01	6	0906	1	10609.60	1.72116e+01	10637	10588
ta01	6	0906	2	11455.80	7.18273e+01	11571	11310
ta01	1	0909	1	10906.80	9.63120e+00	10919	10890
ta01	1	0909	2	14518.60	4.21478e+01	14581	14456
ta01	2	0909	1	19914.30	3.95450e+01	19981	19865
ta01	2	0909	2	18977.60	4.27439e+01	19034	18882
ta01	6	0909	1	10902.30	7.40338e+00	10917	10890
ta01	6	0909	2	14467.90	3.23495e+01	14509	14409
abz7	1	0606	1	9370.50	3.54605e+01	9419	9306
abz7	1	0606	2	10045.50	7.22638e+01	10159	9954
abz7	2	0606	1	15401.30	8.55232e+01	15509	15210
abz7	2	0606	2	13067.60	1.02246e+02	13238	12862
abz7	6	0606	1	9410.20	2.12311e+01	9438	9372
abz7	6	0606	2	10003.00	5.59071e+01	10068	9891
abz7	1	0609	1	10601.70	1.41425e+01	10619	10569
abz7	1	0609	2	12986.70	3.48340e+01	13064	12931
abz7	2	0609	1	16598.90	1.92377e+01	16628	16567
abz7	2	0609	2	16000.60	5.99203e+01	16082	15892
abz7	6	0609	1	10607.90	2.34583e+01	10644	10573
abz7	6	0609	2	13005.50	5.12294e+01	13083	12916
abz7	1	0906	1	10604.00	2.08614e+01	10651	10578
abz7	1	0906	2	11551.80	6.19723e+01	11644	11426
abz7	2	0906	1	19607.70	4.71297e+01	19697	19543
abz7	2	0906	2	15987.40	7.88977e+01	16136	15882
abz7	6	0906	1	10602.30	2.01099e+01	10638	10568
abz7	6	0906	2	11532.50	8.72150e+01	11667	11354
abz7	1	0909	1	10899.20	4.99600e+00	10906	10892
abz7	1	0909	2	14503.30	2.55580e+01	14551	14451
abz7	2	0909	1	19918.00	2.37529e+01	19959	19886
abz7	2	0909	2	19014.60	3.85959e+01	19092	18964
abz7	6	0909	1	10898.30	8.87750e+00	10912	10887
abz7	6	0909	2	14498.60	4.43716e+01	14558	14431

A.3.4 Résultats pour 500 individus après 300 générations

A.3.4.1 Critère $\Pi'_{\min}$

inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	989.00	2.16795e+01	1036	955	ta01	1	0606	1	1357.40	2.26504e+01	1401	1324
ft10	1	0606	2	999.10	2.12389e+01	1047	976	ta01	1	0606	2	1362.10	1.82233e+01	1386	1329
ft10	2	0606	1	1000.20	1.58291e+01	1029	979	ta01	2	0606	1	1370.00	1.18406e+01	1389	1347
ft10	2	0606	2	1004.80	2.30295e+01	1062	973	ta01	2	0606	2	1361.40	1.94689e+01	1392	1332
ft10	6	0606	1	1004.40	1.10291e+01	1024	988	ta01	6	0606	1	1367.60	1.76307e+01	1416	1351
ft10	6	0606	2	1002.00	2.51197e+01	1036	964	ta01	6	0606	2	1357.30	1.72688e+01	1393	1335
ft10	1	0609	1	1050.90	1.72247e+01	1078	1026	ta01	1	0609	1	1395.90	1.69260e+01	1422	1361
ft10	1	0609	2	988.70	1.30158e+01	1015	971	ta01	1	0609	2	1345.40	1.99359e+01	1389	1317
ft10	2	0609	1	1035.10	1.28254e+01	1056	1017	ta01	2	0609	1	1417.20	6.83813e+00	1427	1407
ft10	2	0609	2	991.00	1.67451e+01	1018	955	ta01	2	0609	2	1361.10	1.47882e+01	1380	1333
ft10	6	0609	1	1039.60	1.10923e+01	1058	1018	ta01	6	0609	1	1422.00	1.69823e+01	1452	1401
ft10	6	0609	2	992.10	2.13750e+01	1030	966	ta01	6	0609	2	1367.90	1.38741e+01	1386	1342
ft10	1	0906	1	984.90	1.67657e+01	1016	963	ta01	1	0906	1	1349.80	8.85212e+00	1365	1336
ft10	1	0906	2	986.50	1.43962e+01	1018	969	ta01	1	0906	2	1344.30	1.65412e+01	1378	1324
ft10	2	0906	1	987.40	1.55383e+01	1010	953	ta01	2	0906	1	1370.70	1.66556e+01	1398	1341
ft10	2	0906	2	989.90	1.50496e+01	1011	961	ta01	2	0906	2	1363.90	2.28449e+01	1411	1336
ft10	6	0906	1	989.00	1.80499e+01	1016	964	ta01	6	0906	1	1362.10	1.74152e+01	1405	1341
ft10	6	0906	2	990.40	1.82110e+01	1023	957	ta01	6	0906	2	1361.60	2.01306e+01	1404	1325
ft10	1	0909	1	1040.00	1.63218e+01	1057	1005	ta01	1	0909	1	1407.90	1.54755e+01	1425	1375
ft10	1	0909	2	975.70	1.68822e+01	1010	954	ta01	1	0909	2	1340.30	1.45812e+01	1370	1317
ft10	2	0909	1	1021.00	1.60187e+01	1046	995	ta01	2	0909	1	1407.60	1.10743e+01	1427	1389
ft10	2	0909	2	992.20	1.32121e+01	1021	973	ta01	2	0909	2	1361.30	1.93186e+01	1382	1326
ft10	6	0909	1	1051.60	2.38671e+01	1087	1009	ta01	6	0909	1	1420.90	1.26131e+01	1439	1402
ft10	6	0909	2	976.90	1.94445e+01	1009	936	ta01	6	0909	2	1356.50	6.93181e+00	1367	1345
ft20	1	0606	1	1237.90	1.11665e+01	1257	1223	abz7	1	0606	1	736.10	6.33167e+00	751	728
ft20	1	0606	2	1226.10	1.98970e+01	1247	1178	abz7	1	0606	2	732.30	1.16795e+01	752	712
ft20	2	0606	1	1257.60	2.43401e+01	1286	1213	abz7	2	0606	1	746.80	5.72364e+00	758	740
ft20	2	0606	2	1248.50	1.79067e+01	1272	1205	abz7	2	0606	2	742.20	1.15914e+01	762	719
ft20	6	0606	1	1242.30	1.02669e+01	1257	1222	abz7	6	0606	1	737.40	6.31189e+00	747	726
ft20	6	0606	2	1235.90	1.44184e+01	1256	1203	abz7	6	0606	2	739.80	9.18477e+00	754	720
ft20	1	0609	1	1283.60	2.75144e+01	1329	1241	abz7	1	0609	1	768.30	6.66408e+00	783	759
ft20	1	0609	2	1229.60	9.88130e+00	1254	1216	abz7	1	0609	2	736.10	7.28629e+00	749	724
ft20	2	0609	1	1288.50	1.42355e+01	1313	1266	abz7	2	0609	1	767.80	6.46220e+00	779	759
ft20	2	0609	2	1253.60	1.44305e+01	1281	1233	abz7	2	0609	2	742.60	8.18780e+00	758	728
ft20	6	0609	1	1290.50	1.63844e+01	1314	1260	abz7	6	0609	1	768.30	6.37260e+00	776	756
ft20	6	0609	2	1241.00	1.19248e+01	1255	1214	abz7	6	0609	2	739.50	6.10328e+00	752	730
ft20	1	0906	1	1216.20	1.61295e+01	1250	1197	abz7	1	0906	1	737.10	7.76466e+00	748	724
ft20	1	0906	2	1218.50	2.13319e+01	1251	1185	abz7	1	0906	2	733.10	7.00643e+00	746	724
ft20	2	0906	1	1233.50	1.48273e+01	1254	1213	abz7	2	0906	1	739.90	7.11969e+00	752	726
ft20	2	0906	2	1240.80	1.60424e+01	1263	1212	abz7	2	0906	2	738.30	1.11000e+01	760	720
ft20	6	0906	1	1229.00	1.71988e+01	1246	1201	abz7	6	0906	1	741.40	7.98999e+00	753	727
ft20	6	0906	2	1223.00	2.06833e+01	1254	1190	abz7	6	0906	2	735.10	9.76166e+00	749	719
ft20	1	0909	1	1265.50	1.38076e+01	1285	1245	abz7	1	0909	1	765.20	4.83322e+00	774	760
ft20	1	0909	2	1215.80	1.10345e+01	1232	1199	abz7	1	0909	2	728.50	7.52662e+00	743	718
ft20	2	0909	1	1278.10	1.65919e+01	1314	1255	abz7	2	0909	1	764.30	4.62709e+00	771	753
ft20	2	0909	2	1236.80	2.07017e+01	1270	1198	abz7	2	0909	2	737.20	8.47113e+00	751	724
ft20	6	0909	1	1278.90	1.94394e+01	1305	1235	abz7	6	0909	1	769.60	4.90306e+00	776	759
ft20	6	0909	2	1225.50	2.24288e+01	1262	1191	abz7	6	0909	2	739.70	9.34933e+00	755	720
inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min

A.3.4.2 Critère $\Pi'_{\max}$

inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	990.80	2.17154e+01	1036	955	ta01	1	0606	1	1365.70	2.21993e+01	1401	1335
ft10	1	0606	2	1000.90	2.47849e+01	1062	976	ta01	1	0606	2	1362.30	1.79000e+01	1386	1329
ft10	2	0606	1	1000.60	1.60200e+01	1029	979	ta01	2	0606	1	1373.80	1.14525e+01	1390	1350
ft10	2	0606	2	1005.20	2.29076e+01	1062	973	ta01	2	0606	2	1361.80	1.92135e+01	1392	1332
ft10	6	0606	1	1005.70	1.16880e+01	1024	988	ta01	6	0606	1	1381.60	1.55061e+01	1416	1360
ft10	6	0606	2	1002.10	2.51851e+01	1036	964	ta01	6	0606	2	1359.20	1.58795e+01	1393	1339
ft10	1	0609	1	1104.60	2.05679e+01	1134	1073	ta01	1	0609	1	1451.00	1.41492e+01	1472	1424
ft10	1	0609	2	988.70	1.30158e+01	1015	971	ta01	1	0609	2	1345.90	1.97051e+01	1389	1317
ft10	2	0609	1	1095.00	2.01147e+01	1119	1062	ta01	2	0609	1	1459.40	1.25634e+01	1477	1435
ft10	2	0609	2	991.00	1.67451e+01	1018	955	ta01	2	0609	2	1363.80	1.19817e+01	1380	1343
ft10	6	0609	1	1119.60	1.03846e+01	1132	1104	ta01	6	0609	1	1485.40	1.04614e+01	1501	1464
ft10	6	0609	2	994.00	1.97129e+01	1030	973	ta01	6	0609	2	1373.60	1.33207e+01	1386	1346
ft10	1	0906	1	995.60	1.21836e+01	1016	976	ta01	1	0906	1	1364.70	9.40266e+00	1380	1344
ft10	1	0906	2	986.80	1.46615e+01	1018	969	ta01	1	0906	2	1345.60	1.60823e+01	1378	1324
ft10	2	0906	1	991.60	1.37782e+01	1012	962	ta01	2	0906	1	1382.90	1.91439e+01	1411	1354
ft10	2	0906	2	989.90	1.50496e+01	1011	961	ta01	2	0906	2	1364.30	2.26762e+01	1411	1336
ft10	6	0906	1	1015.10	1.08945e+01	1030	993	ta01	6	0906	1	1395.30	9.50842e+00	1416	1379
ft10	6	0906	2	992.50	1.97952e+01	1023	959	ta01	6	0906	2	1364.00	1.97028e+01	1404	1327
ft10	1	0909	1	1115.20	2.40408e+01	1160	1079	ta01	1	0909	1	1482.40	1.06977e+01	1511	1471
ft10	1	0909	2	977.20	1.61481e+01	1013	958	ta01	1	0909	2	1344.80	1.27106e+01	1370	1319
ft10	2	0909	1	1102.20	1.55100e+01	1126	1067	ta01	2	0909	1	1479.20	9.52680e+00	1495	1462
ft10	2	0909	2	992.90	1.33375e+01	1021	973	ta01	2	0909	2	1364.00	2.12368e+01	1388	1326
ft10	6	0909	1	1157.30	1.61186e+01	1179	1132	ta01	6	0909	1	1532.20	1.02352e+01	1552	1514
ft10	6	0909	2	981.40	1.71884e+01	1009	942	ta01	6	0909	2	1362.50	8.59360e+00	1375	1351
ft20	1	0606	1	1246.70	9.65453e+00	1262	1234	abz7	1	0606	1	740.10	5.97411e+00	755	734
ft20	1	0606	2	1226.30	1.96929e+01	1247	1178	abz7	1	0606	2	732.40	1.18507e+01	753	712
ft20	2	0606	1	1262.60	2.56990e+01	1289	1214	abz7	2	0606	1	748.70	7.34915e+00	765	740
ft20	2	0606	2	1251.20	1.24643e+01	1272	1231	abz7	2	0606	2	742.50	1.15607e+01	762	719
ft20	6	0606	1	1249.00	9.13236e+00	1261	1228	abz7	6	0606	1	741.60	7.33757e+00	757	728
ft20	6	0606	2	1238.30	1.45193e+01	1256	1203	abz7	6	0606	2	740.90	9.43875e+00	754	721
ft20	1	0609	1	1345.80	2.43138e+01	1376	1307	abz7	1	0609	1	793.20	4.95580e+00	805	787
ft20	1	0609	2	1232.90	9.70000e+00	1254	1218	abz7	1	0609	2	737.10	6.99214e+00	749	725
ft20	2	0609	1	1337.60	1.98655e+01	1364	1298	abz7	2	0609	1	790.80	6.36867e+00	802	779
ft20	2	0609	2	1255.40	1.51605e+01	1288	1233	abz7	2	0609	2	743.40	8.55804e+00	758	728
ft20	6	0609	1	1351.00	2.11518e+01	1387	1313	abz7	6	0609	1	796.50	8.35763e+00	810	782
ft20	6	0609	2	1243.80	1.36514e+01	1271	1218	abz7	6	0609	2	743.00	5.25357e+00	754	737
ft20	1	0906	1	1233.10	1.37510e+01	1254	1210	abz7	1	0906	1	747.30	7.37631e+00	760	733
ft20	1	0906	2	1219.80	2.02079e+01	1251	1188	abz7	1	0906	2	734.10	7.30000e+00	749	725
ft20	2	0906	1	1239.90	1.35089e+01	1257	1219	abz7	2	0906	1	745.90	8.23954e+00	760	730
ft20	2	0906	2	1240.80	1.60424e+01	1263	1212	abz7	2	0906	2	739.40	1.04326e+01	760	725
ft20	6	0906	1	1243.10	1.48691e+01	1266	1224	abz7	6	0906	1	757.70	7.86193e+00	775	742
ft20	6	0906	2	1224.90	1.92273e+01	1254	1195	abz7	6	0906	2	736.30	1.00504e+01	751	719
ft20	1	0909	1	1344.00	2.44908e+01	1392	1317	abz7	1	0909	1	802.60	4.75815e+00	814	797
ft20	1	0909	2	1219.40	1.28390e+01	1239	1199	abz7	1	0909	2	732.10	7.36817e+00	744	721
ft20	2	0909	1	1345.10	1.65557e+01	1374	1312	abz7	2	0909	1	798.80	3.18748e+00	803	792
ft20	2	0909	2	1238.20	1.88563e+01	1271	1205	abz7	2	0909	2	740.20	8.03492e+00	751	724
ft20	6	0909	1	1366.60	2.50448e+01	1392	1306	abz7	6	0909	1	817.20	5.34416e+00	825	811
ft20	6	0909	2	1230.40	2.15323e+01	1265	1194	abz7	6	0909	2	744.90	8.85946e+00	759	727
inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min

A.3.4.3 Critère $\Pi'_{\mathbf{moy}}$

inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	990.10	2.31881e+01	1040.00	955.00
ft10	1	0606	2	1001.10	2.48695e+01	1060.00	976.00
ft10	2	0606	1	1000.20	1.62099e+01	1030.00	979.00
ft10	2	0606	2	1004.90	2.25940e+01	1060.00	973.00
ft10	6	0606	1	1004.40	1.07257e+01	1020.00	988.00
ft10	6	0606	2	1002.10	2.51732e+01	1040.00	964.00
ft10	1	0609	1	1070.00	2.04939e+01	1100.00	1040.00
ft10	1	0609	2	988.90	1.38018e+01	1020.00	971.00
ft10	2	0609	1	1057.00	1.67631e+01	1090.00	1030.00
ft10	2	0609	2	991.10	1.70203e+01	1020.00	955.00
ft10	6	0609	1	1069.00	1.04403e+01	1090.00	1050.00
ft10	6	0609	2	993.60	2.01554e+01	1030.00	973.00
ft10	1	0906	1	987.00	1.68345e+01	1020.00	963.00
ft10	1	0906	2	986.80	1.49184e+01	1020.00	969.00
ft10	2	0906	1	988.60	1.54221e+01	1010.00	954.00
ft10	2	0906	2	988.80	1.40414e+01	1010.00	961.00
ft10	6	0906	1	1000.00	1.32665e+01	1020.00	972.00
ft10	6	0906	2	991.20	1.92499e+01	1020.00	957.00
ft10	1	0909	1	1068.00	1.60000e+01	1090.00	1040.00
ft10	1	0909	2	976.60	1.58442e+01	1010.00	956.00
ft10	2	0909	1	1048.00	1.66132e+01	1070.00	1020.00
ft10	2	0909	2	992.60	1.34996e+01	1020.00	973.00
ft10	6	0909	1	1096.00	1.85472e+01	1120.00	1060.00
ft10	6	0909	2	979.60	1.77888e+01	1010.00	938.00
ft20	1	0606	1	1244.00	1.01980e+01	1260.00	1230.00
ft20	1	0606	2	1228.00	1.93907e+01	1250.00	1180.00
ft20	2	0606	1	1258.00	2.60000e+01	1290.00	1210.00
ft20	2	0606	2	1249.00	1.51327e+01	1270.00	1220.00
ft20	6	0606	1	1247.00	7.81025e+00	1260.00	1230.00
ft20	6	0606	2	1238.00	1.66132e+01	1260.00	1200.00
ft20	1	0609	1	1305.00	2.72947e+01	1350.00	1260.00
ft20	1	0609	2	1230.00	7.74597e+00	1250.00	1220.00
ft20	2	0609	1	1310.00	2.00000e+01	1330.00	1270.00
ft20	2	0609	2	1254.00	1.68523e+01	1290.00	1230.00
ft20	6	0609	1	1318.00	2.27156e+01	1360.00	1280.00
ft20	6	0609	2	1241.00	1.04403e+01	1260.00	1220.00
ft20	1	0906	1	1222.00	1.72047e+01	1250.00	1200.00
ft20	1	0906	2	1220.00	2.09762e+01	1250.00	1190.00
ft20	2	0906	1	1235.00	1.36015e+01	1250.00	1210.00
ft20	2	0906	2	1239.00	1.70000e+01	1260.00	1210.00
ft20	6	0906	1	1234.00	1.62481e+01	1250.00	1210.00
ft20	6	0906	2	1224.00	2.10713e+01	1250.00	1190.00
ft20	1	0909	1	1300.00	2.09762e+01	1340.00	1280.00
ft20	1	0909	2	1218.00	1.24900e+01	1240.00	1200.00
ft20	2	0909	1	1305.00	1.50000e+01	1330.00	1280.00
ft20	2	0909	2	1238.00	1.93907e+01	1270.00	1200.00
ft20	6	0909	1	1314.00	2.41661e+01	1340.00	1270.00
ft20	6	0909	2	1227.00	2.19317e+01	1260.00	1190.00

inst	op	taux	sch	moy	écart	max	min
ta01	1	0606	1	1363.00	2.28254e+01	1400.00	1330.00
ta01	1	0606	2	1362.00	1.93907e+01	1390.00	1330.00
ta01	2	0606	1	1374.00	1.11355e+01	1390.00	1350.00
ta01	2	0606	2	1361.00	1.92094e+01	1390.00	1330.00
ta01	6	0606	1	1375.00	1.74642e+01	1420.00	1350.00
ta01	6	0606	2	1360.00	1.54919e+01	1390.00	1340.00
ta01	1	0609	1	1420.00	1.73205e+01	1450.00	1400.00
ta01	1	0609	2	1346.00	1.90788e+01	1390.00	1320.00
ta01	2	0609	1	1430.00	7.74597e+00	1440.00	1420.00
ta01	2	0609	2	1362.00	1.24900e+01	1380.00	1340.00
ta01	6	0609	1	1446.00	1.35647e+01	1460.00	1420.00
ta01	6	0609	2	1372.00	1.40000e+01	1390.00	1350.00
ta01	1	0906	1	1357.00	7.81025e+00	1370.00	1340.00
ta01	1	0906	2	1345.00	1.68819e+01	1380.00	1320.00
ta01	2	0906	1	1378.00	1.77764e+01	1410.00	1350.00
ta01	2	0906	2	1365.00	2.15639e+01	1410.00	1340.00
ta01	6	0906	1	1373.00	1.61555e+01	1410.00	1360.00
ta01	6	0906	2	1363.00	1.84662e+01	1400.00	1330.00
ta01	1	0909	1	1434.00	1.01980e+01	1450.00	1420.00
ta01	1	0909	2	1342.00	1.32665e+01	1370.00	1320.00
ta01	2	0909	1	1433.00	1.10000e+01	1450.00	1420.00
ta01	2	0909	2	1362.00	1.88680e+01	1380.00	1330.00
ta01	6	0909	1	1465.00	8.06226e+00	1480.00	1450.00
ta01	6	0909	2	1360.00	7.74597e+00	1370.00	1350.00
abz7	1	0606	1	738.00	6.43428e+00	753.00	728.00
abz7	1	0606	2	732.40	1.18507e+01	753.00	712.00
abz7	2	0606	1	747.70	6.34114e+00	761.00	740.00
abz7	2	0606	2	742.40	1.15343e+01	762.00	719.00
abz7	6	0606	1	739.90	6.62495e+00	751.00	727.00
abz7	6	0606	2	740.10	9.07138e+00	754.00	721.00
abz7	1	0609	1	777.80	4.23792e+00	787.00	772.00
abz7	1	0609	2	736.80	7.03989e+00	749.00	725.00
abz7	2	0609	1	775.50	4.92443e+00	785.00	769.00
abz7	2	0609	2	743.00	8.30662e+00	758.00	728.00
abz7	6	0609	1	779.40	4.73709e+00	787.00	771.00
abz7	6	0609	2	741.60	5.80000e+00	754.00	731.00
abz7	1	0906	1	741.50	8.38153e+00	755.00	728.00
abz7	1	0906	2	733.70	7.47061e+00	749.00	724.00
abz7	2	0906	1	742.40	7.04557e+00	752.00	729.00
abz7	2	0906	2	738.80	1.11786e+01	760.00	720.00
abz7	6	0906	1	748.40	7.15821e+00	764.00	737.00
abz7	6	0906	2	735.50	9.75961e+00	749.00	719.00
abz7	1	0909	1	780.80	4.30813e+00	787.00	773.00
abz7	1	0909	2	730.10	7.32735e+00	743.00	719.00
abz7	2	0909	1	779.00	4.89898e+00	785.00	771.00
abz7	2	0909	2	738.50	7.97810e+00	751.00	724.00
abz7	6	0909	1	789.70	3.31813e+00	794.00	786.00
abz7	6	0909	2	742.60	8.18780e+00	756.00	725.00

A.3.4.4 Critère $\Pi'_{\text{écart}}$

inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	0.25	4.79217e-01	1.610	0.000
ft10	1	0606	2	0.32	6.77182e-01	2.110	0.000
ft10	2	0606	1	0.05	1.50600e-01	0.502	0.000
ft10	2	0606	2	0.02	5.37000e-02	0.179	0.000
ft10	6	0606	1	0.33	5.61467e-01	1.740	0.000
ft10	6	0606	2	0.01	1.89600e-02	0.063	0.000
ft10	1	0609	1	7.89	2.05295e+00	11.900	4.540
ft10	1	0609	2	0.00	0.00000e+00	0.000	0.000
ft10	2	0609	1	8.17	3.73083e+00	14.900	1.500
ft10	2	0609	2	0.00	0.00000e+00	0.000	0.000
ft10	6	0609	1	12.35	2.59709e+00	17.000	8.840
ft10	6	0609	2	0.21	4.31071e-01	1.210	0.000
ft10	1	0906	1	1.59	1.57334e+00	5.350	0.000
ft10	1	0906	2	0.14	4.23000e-01	1.410	0.000
ft10	2	0906	1	0.69	5.89031e-01	1.710	0.000
ft10	2	0906	2	0.00	0.00000e+00	0.000	0.000
ft10	6	0906	1	5.22	2.83755e+00	10.000	0.690
ft10	6	0906	2	0.19	3.84938e-01	1.200	0.000
ft10	1	0909	1	12.95	2.32835e+00	17.100	9.850
ft10	1	0909	2	0.42	6.51809e-01	1.970	0.000
ft10	2	0909	1	10.64	1.75773e+00	13.200	7.680
ft10	2	0909	2	0.13	2.94519e-01	0.994	0.000
ft10	6	0909	1	16.62	3.00493e+00	21.300	12.200
ft10	6	0909	2	1.20	2.10903e+00	6.790	0.000
ft20	1	0606	1	1.05	8.99711e-01	2.700	0.000
ft20	1	0606	2	0.08	2.34600e-01	0.782	0.000
ft20	2	0606	1	0.82	2.08470e+00	7.040	0.000
ft20	2	0606	2	0.27	6.92099e-01	2.320	0.000
ft20	6	0606	1	1.16	1.20633e+00	3.800	0.000
ft20	6	0606	2	0.40	6.82700e-01	1.880	0.000
ft20	1	0609	1	11.01	4.12475e+00	20.600	5.140
ft20	1	0609	2	0.43	5.37841e-01	1.730	0.000
ft20	2	0609	1	7.75	2.40532e+00	12.900	4.130
ft20	2	0609	2	0.53	8.28939e-01	2.540	0.000
ft20	6	0609	1	9.35	3.75419e+00	18.100	4.200
ft20	6	0609	2	0.38	4.30808e-01	1.050	0.000
ft20	1	0906	1	2.89	1.22693e+00	4.860	0.179
ft20	1	0906	2	0.17	2.65401e-01	0.673	0.000
ft20	2	0906	1	0.89	9.70853e-01	3.180	0.000
ft20	2	0906	2	0.00	0.00000e+00	0.000	0.000
ft20	6	0906	1	2.05	9.35395e-01	3.650	0.341
ft20	6	0906	2	0.36	4.41430e-01	1.210	0.000
ft20	1	0909	1	12.25	2.39374e+00	15.100	7.350
ft20	1	0909	2	0.91	7.84382e-01	2.320	0.000
ft20	2	0909	1	10.28	2.60466e+00	16.800	7.510
ft20	2	0909	2	0.54	1.55965e+00	5.220	0.000
ft20	6	0909	1	13.93	2.31854e+00	17.100	9.710
ft20	6	0909	2	1.05	1.65123e+00	5.820	0.000

inst	op	taux	sch	moy	écart	max	min
ta01	1	0606	1	0.97	8.32022e-01	2.710	0.000
ta01	1	0606	2	0.07	2.13900e-01	0.713	0.000
ta01	2	0606	1	0.52	9.57525e-01	3.100	0.000
ta01	2	0606	2	0.10	2.40293e-01	0.810	0.000
ta01	6	0606	1	2.84	2.13256e+00	6.570	0.000
ta01	6	0606	2	0.32	5.02485e-01	1.370	0.000
ta01	1	0609	1	8.23	2.93442e+00	14.400	4.080
ta01	1	0609	2	0.14	2.78825e-01	0.767	0.000
ta01	2	0609	1	6.22	2.40550e+00	9.540	2.630
ta01	2	0609	2	0.49	1.00354e+00	3.340	0.000
ta01	6	0609	1	9.19	2.95288e+00	14.800	3.600
ta01	6	0609	2	0.88	7.13248e-01	2.010	0.000
ta01	1	0906	1	2.40	1.47093e+00	5.390	0.000
ta01	1	0906	2	0.25	4.41535e-01	1.390	0.000
ta01	2	0906	1	2.09	1.36614e+00	3.950	0.000
ta01	2	0906	2	0.15	4.45709e-01	1.490	0.000
ta01	6	0906	1	5.28	2.55861e+00	8.900	2.410
ta01	6	0906	2	0.38	4.60558e-01	1.220	0.000
ta01	1	0909	1	12.01	4.02719e+00	21.000	7.470
ta01	1	0909	2	0.87	9.78437e-01	3.270	0.000
ta01	2	0909	1	10.72	1.32748e+00	12.700	9.000
ta01	2	0909	2	0.60	6.04377e-01	1.800	0.000
ta01	6	0909	1	17.26	2.06310e+00	21.300	14.300
ta01	6	0909	2	0.90	8.35396e-01	2.970	0.000
abz7	1	0606	1	0.63	2.95740e-01	1.150	0.244
abz7	1	0606	2	0.04	1.27200e-01	0.424	0.000
abz7	2	0606	1	0.44	7.20565e-01	2.060	0.000
abz7	2	0606	2	0.14	2.99909e-01	0.946	0.000
abz7	6	0606	1	0.66	8.13686e-01	2.800	0.000
abz7	6	0606	2	0.28	4.64450e-01	1.460	0.000
abz7	1	0609	1	3.41	1.13009e+00	5.320	2.050
abz7	1	0609	2	0.19	2.75142e-01	0.855	0.000
abz7	2	0609	1	3.43	1.34890e+00	5.760	1.440
abz7	2	0609	2	0.16	4.09024e-01	1.380	0.000
abz7	6	0609	1	4.19	8.83292e-01	5.720	2.530
abz7	6	0609	2	0.82	1.25360e+00	4.340	0.000
abz7	1	0906	1	1.75	4.28445e-01	2.580	1.180
abz7	1	0906	2	0.29	5.37829e-01	1.820	0.000
abz7	2	0906	1	1.10	7.47989e-01	3.140	0.411
abz7	2	0906	2	0.15	2.84562e-01	0.923	0.000
abz7	6	0906	1	1.98	7.93688e-01	3.390	0.790
abz7	6	0906	2	0.16	2.44729e-01	0.725	0.000
abz7	1	0909	1	5.80	1.22250e+00	7.430	3.780
abz7	1	0909	2	0.75	7.06229e-01	2.290	0.000
abz7	2	0909	1	5.44	8.94336e-01	7.110	4.090
abz7	2	0909	2	0.56	5.58928e-01	1.900	0.000
abz7	6	0909	1	7.25	1.08535e+00	9.840	5.770
abz7	6	0909	2	0.67	3.95074e-01	1.370	0.000

A.3.4.5 Critère $\Pi'_{\text{éval}}$

inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	126964.80	1.59313e+02	127220	126643	ta01	1	0606	1	126948.10	1.55183e+02	127161	126642
ft10	1	0606	2	135960.70	3.13487e+02	136576	135411	ta01	1	0606	2	135873.00	1.71894e+02	136178	135673
ft10	2	0606	1	217078.00	3.00505e+02	217527	216445	ta01	2	0606	1	217003.20	3.05729e+02	217356	216427
ft10	2	0606	2	181078.80	2.97131e+02	181590	180476	ta01	2	0606	2	180887.20	2.57337e+02	181296	180384
ft10	6	0606	1	127036.10	1.44850e+02	127256	126760	ta01	6	0606	1	126963.70	1.33052e+02	127119	126710
ft10	6	0606	2	135988.60	2.27017e+02	136512	135710	ta01	6	0606	2	136283.70	2.48002e+02	136873	135955
ft10	1	0609	1	145036.50	1.01791e+02	145178	144869	ta01	1	0609	1	144969.60	6.26725e+01	145073	144844
ft10	1	0609	2	180958.60	2.24237e+02	181359	180566	ta01	1	0609	2	181060.00	1.21761e+02	181251	180814
ft10	2	0609	1	235007.40	2.07684e+02	235378	234626	ta01	2	0609	1	235006.20	1.65119e+02	235302	234680
ft10	2	0609	2	226016.60	2.57603e+02	226428	225558	ta01	2	0609	2	226019.40	3.29921e+02	226462	225542
ft10	6	0609	1	144987.40	9.30937e+01	145138	144776	ta01	6	0609	1	145049.80	7.82136e+01	145202	144936
ft10	6	0609	2	180930.60	1.70986e+02	181292	180729	ta01	6	0609	2	181087.30	1.82571e+02	181474	180806
ft10	1	0906	1	145000.60	6.42887e+01	145148	144895	ta01	1	0906	1	145029.30	7.35623e+01	145137	144913
ft10	1	0906	2	158416.20	2.54140e+02	158783	158108	ta01	1	0906	2	158383.50	3.26584e+02	158975	157681
ft10	2	0906	1	280091.90	1.60546e+02	280365	279852	ta01	2	0906	1	280053.70	9.32427e+01	280225	279900
ft10	2	0906	2	225880.80	2.33963e+02	226226	225444	ta01	2	0906	2	226055.80	3.37191e+02	226712	225528
ft10	6	0906	1	144978.90	4.15053e+01	145023	144893	ta01	6	0906	1	145015.10	9.91054e+01	145187	144855
ft10	6	0906	2	158626.70	2.93626e+02	159026	158083	ta01	6	0906	2	158473.80	3.87164e+02	159106	157965
ft10	1	0909	1	149522.30	4.67740e+01	149617	149441	ta01	1	0909	1	149499.50	3.03653e+01	149542	149453
ft10	1	0909	2	203421.20	1.52146e+02	203628	203146	ta01	1	0909	2	203472.20	1.80779e+02	203751	203191
ft10	2	0909	1	284550.20	9.19791e+01	284652	284341	ta01	2	0909	1	284546.00	1.27635e+02	284672	284295
ft10	2	0909	2	271037.20	2.21905e+02	271430	270746	ta01	2	0909	2	271008.40	2.26466e+02	271306	270546
ft10	6	0909	1	149483.90	4.73211e+01	149564	149405	ta01	6	0909	1	149518.70	2.83515e+01	149559	149472
ft10	6	0909	2	203581.80	1.60969e+02	203813	203297	ta01	6	0909	2	203492.30	1.39782e+02	203766	203260
ft20	1	0606	1	127025.40	1.08816e+02	127227	126890	abz7	1	0606	1	126988.80	1.26033e+02	127231	126764
ft20	1	0606	2	135930.70	3.67280e+02	136579	135524	abz7	1	0606	2	136148.50	3.22808e+02	136720	135678
ft20	2	0606	1	216953.60	1.49490e+02	217266	216793	abz7	2	0606	1	217083.50	2.95201e+02	217608	216522
ft20	2	0606	2	180976.00	2.79467e+02	181372	180558	abz7	2	0606	2	181246.60	3.26622e+02	181780	180614
ft20	6	0606	1	127091.30	1.80939e+02	127400	126747	abz7	6	0606	1	127044.10	1.42781e+02	127343	126870
ft20	6	0606	2	136078.40	2.74300e+02	136444	135590	abz7	6	0606	2	136047.10	3.30318e+02	136724	135532
ft20	1	0609	1	145042.70	7.70961e+01	145191	144937	abz7	1	0609	1	144977.90	8.69994e+01	145089	144873
ft20	1	0609	2	180994.00	2.38430e+02	181557	180601	abz7	1	0609	2	181034.30	2.32363e+02	181342	180627
ft20	2	0609	1	234974.50	1.09643e+02	235142	234832	abz7	2	0609	1	235002.40	2.87505e+02	235591	234545
ft20	2	0609	2	225937.80	1.63676e+02	226144	225552	abz7	2	0609	2	225971.40	3.11154e+02	226504	225394
ft20	6	0609	1	144952.20	5.38717e+01	145020	144859	abz7	6	0609	1	144984.20	9.95849e+01	145183	144817
ft20	6	0609	2	180857.20	2.34298e+02	181292	180574	abz7	6	0609	2	181032.30	1.82937e+02	181379	180777
ft20	1	0906	1	144980.10	1.25296e+02	145213	144811	abz7	1	0906	1	145012.50	7.33597e+01	145124	144892
ft20	1	0906	2	158530.80	2.17243e+02	158865	158030	abz7	1	0906	2	158526.80	3.56362e+02	159015	157927
ft20	2	0906	1	279972.50	2.12943e+02	280302	279654	abz7	2	0906	1	280044.80	1.48370e+02	280261	279842
ft20	2	0906	2	226178.60	2.52137e+02	226602	225728	abz7	2	0906	2	226053.60	2.34203e+02	226420	225642
ft20	6	0906	1	145000.00	8.33103e+01	145139	144893	abz7	6	0906	1	145018.50	6.07902e+01	145109	144901
ft20	6	0906	2	158584.40	2.88276e+02	159034	158007	abz7	6	0906	2	158436.80	2.09613e+02	158804	158064
ft20	1	0909	1	149455.30	3.81760e+01	149501	149393	abz7	1	0909	1	149501.40	2.72918e+01	149552	149462
ft20	1	0909	2	203520.30	1.54810e+02	203735	203255	abz7	1	0909	2	203431.90	1.50088e+02	203761	203203
ft20	2	0909	1	284476.60	8.08557e+01	284601	284339	abz7	2	0909	1	284488.00	1.67117e+02	284707	284230
ft20	2	0909	2	271051.00	2.38951e+02	271580	270744	abz7	2	0909	2	270900.80	2.57904e+02	271234	270522
ft20	6	0909	1	149492.60	4.45066e+01	149551	149408	abz7	6	0909	1	149485.40	5.27014e+01	149553	149369
ft20	6	0909	2	203475.90	2.52672e+02	203883	203040	abz7	6	0909	2	203395.70	1.73224e+02	203801	203133
inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min

A.3.5 Résultats après une génération pour 500 individus guidés par f_2

La fonction multicritère f_2 est effectivement utilisée pour guider l'AE, les statistiques sur le makespan sont calculées directement par l'AE en ce qui concerne la colonne min, mais elles sont

estimées après exécution de l'AE en ce qui concerne la colonne moyenne.

A.3.5.1 Critère $\Pi'_{\min}$

inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	1195.90	2.10545e+01	1227	1150	ta01	1	0606	1	1548.20	1.80931e+01	1581	1516
ft10	1	0606	2	1210.50	1.75912e+01	1245	1188	ta01	1	0606	2	1601.70	1.84502e+01	1629	1569
ft10	2	0606	1	1198.80	2.45308e+01	1250	1162	ta01	2	0606	1	1539.60	1.88637e+01	1573	1511
ft10	2	0606	2	1199.30	1.99852e+01	1234	1170	ta01	2	0606	2	1601.30	1.75673e+01	1630	1564
ft10	6	0606	1	1233.50	1.91220e+01	1259	1192	ta01	6	0606	1	1552.80	2.64265e+01	1588	1504
ft10	6	0606	2	1204.90	2.71273e+01	1238	1139	ta01	6	0606	2	1626.80	1.75203e+01	1642	1587
ft10	1	0609	1	1227.50	2.51803e+01	1249	1167	ta01	1	0609	1	1548.40	1.11911e+01	1563	1532
ft10	1	0609	2	1196.90	3.15387e+01	1239	1142	ta01	1	0609	2	1593.40	2.85068e+01	1630	1545
ft10	2	0609	1	1234.80	3.78360e+01	1293	1180	ta01	2	0609	1	1537.40	2.14672e+01	1571	1496
ft10	2	0609	2	1194.30	3.75847e+01	1238	1117	ta01	2	0609	2	1585.80	3.29084e+01	1628	1530
ft10	6	0609	1	1248.60	2.39758e+01	1277	1208	ta01	6	0609	1	1561.10	1.07745e+01	1578	1540
ft10	6	0609	2	1207.00	2.69629e+01	1255	1173	ta01	6	0609	2	1610.70	2.01547e+01	1642	1579
ft10	1	0906	1	1201.50	2.07087e+01	1236	1172	ta01	1	0906	1	1531.10	1.92377e+01	1556	1497
ft10	1	0906	2	1197.00	2.58612e+01	1228	1132	ta01	1	0906	2	1592.50	3.16741e+01	1642	1538
ft10	2	0906	1	1191.30	3.10871e+01	1223	1140	ta01	2	0906	1	1543.90	1.49830e+01	1562	1508
ft10	2	0906	2	1174.10	3.03396e+01	1215	1127	ta01	2	0906	2	1600.10	2.47889e+01	1639	1559
ft10	6	0906	1	1182.60	3.59700e+01	1250	1111	ta01	6	0906	1	1556.20	1.52237e+01	1571	1521
ft10	6	0906	2	1189.90	2.24074e+01	1235	1156	ta01	6	0906	2	1610.50	1.81948e+01	1642	1571
ft10	1	0909	1	1213.20	2.00240e+01	1249	1182	ta01	1	0909	1	1547.90	1.22511e+01	1568	1524
ft10	1	0909	2	1169.00	3.15500e+01	1227	1118	ta01	1	0909	2	1596.60	2.25442e+01	1621	1540
ft10	2	0909	1	1213.70	3.25701e+01	1265	1163	ta01	2	0909	1	1542.00	7.94984e+00	1553	1531
ft10	2	0909	2	1197.50	3.19758e+01	1255	1154	ta01	2	0909	2	1590.80	3.62155e+01	1642	1533
ft10	6	0909	1	1236.00	2.27772e+01	1263	1186	ta01	6	0909	1	1555.60	2.05728e+01	1580	1514
ft10	6	0909	2	1196.80	3.87061e+01	1232	1106	ta01	6	0909	2	1595.30	2.86288e+01	1629	1531
ft20	1	0606	1	1438.70	2.18817e+01	1480	1412	abz7	1	0606	1	832.30	7.22565e+00	843	814
ft20	1	0606	2	1432.10	3.37119e+01	1473	1359	abz7	1	0606	2	860.20	1.72093e+01	877	820
ft20	2	0606	1	1444.70	3.27843e+01	1495	1400	abz7	2	0606	1	833.10	5.95735e+00	841	824
ft20	2	0606	2	1427.40	2.55898e+01	1455	1363	abz7	2	0606	2	858.70	1.04790e+01	877	845
ft20	6	0606	1	1427.30	2.71737e+01	1470	1376	abz7	6	0606	1	839.80	7.69155e+00	849	829
ft20	6	0606	2	1447.60	2.40591e+01	1493	1413	abz7	6	0606	2	873.20	8.26801e+00	877	849
ft20	1	0609	1	1459.80	2.47984e+01	1499	1415	abz7	1	0609	1	828.30	5.83181e+00	838	818
ft20	1	0609	2	1444.80	2.35364e+01	1489	1400	abz7	1	0609	2	856.20	1.76624e+01	877	824
ft20	2	0609	1	1452.20	3.24648e+01	1510	1390	abz7	2	0609	1	829.80	7.88416e+00	842	816
ft20	2	0609	2	1448.70	3.04304e+01	1483	1387	abz7	2	0609	2	864.30	7.92528e+00	873	845
ft20	6	0609	1	1457.40	2.22090e+01	1503	1415	abz7	6	0609	1	840.60	5.16140e+00	848	831
ft20	6	0609	2	1449.50	2.41919e+01	1483	1389	abz7	6	0609	2	865.50	9.03604e+00	877	854
ft20	1	0906	1	1443.40	1.73159e+01	1461	1412	abz7	1	0906	1	826.60	7.11618e+00	838	813
ft20	1	0906	2	1438.40	1.90484e+01	1463	1397	abz7	1	0906	2	850.80	1.42534e+01	864	824
ft20	2	0906	1	1419.40	2.93912e+01	1461	1365	abz7	2	0906	1	831.20	6.79412e+00	841	817
ft20	2	0906	2	1430.70	2.27423e+01	1461	1392	abz7	2	0906	2	856.80	9.95791e+00	870	840
ft20	6	0906	1	1438.60	2.21233e+01	1472	1410	abz7	6	0906	1	838.40	6.62118e+00	849	829
ft20	6	0906	2	1431.00	2.56671e+01	1479	1390	abz7	6	0906	2	858.80	1.39771e+01	877	836
ft20	1	0909	1	1460.40	2.06553e+01	1483	1415	abz7	1	0909	1	828.20	6.61513e+00	841	819
ft20	1	0909	2	1432.40	2.17035e+01	1466	1391	abz7	1	0909	2	856.00	1.18237e+01	870	829
ft20	2	0909	1	1469.10	3.45006e+01	1515	1415	abz7	2	0909	1	825.00	6.32456e+00	837	814
ft20	2	0909	2	1431.10	1.64466e+01	1455	1403	abz7	2	0909	2	857.90	1.13088e+01	876	840
ft20	6	0909	1	1464.30	2.67434e+01	1517	1415	abz7	6	0909	1	836.20	6.52380e+00	845	827
ft20	6	0909	2	1423.30	1.88629e+01	1462	1384	abz7	6	0909	2	856.00	1.49265e+01	877	825
inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min

A.3.5.2 Critère $\Pi'_{\mathbf{moy}}$

inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	1434.50	5.38615e+00	1443.01	1425.98	ta01	1	0606	1	1794.45	1.32252e+01	1814.38	1772.43
ft10	1	0606	2	1477.49	7.23659e+00	1489.83	1468.54	ta01	1	0606	2	1930.79	1.32250e+01	1966.45	1914.01
ft10	2	0606	1	1436.20	3.40500e+00	1443.01	1430.24	ta01	2	0606	1	1802.32	1.12561e+01	1814.38	1782.92
ft10	2	0606	2	1476.21	6.25736e+00	1485.58	1464.29	ta01	2	0606	2	1937.61	1.28972e+01	1955.96	1908.77
ft10	6	0606	1	1444.28	5.40100e+00	1451.52	1434.49	ta01	6	0606	1	1821.19	1.40812e+01	1835.35	1782.91
ft10	6	0606	2	1480.04	7.86167e+00	1494.09	1464.29	ta01	6	0606	2	1937.08	9.72501e+00	1945.47	1914.01
ft10	1	0609	1	1409.38	1.27701e+00	1413.21	1408.95	ta01	1	0609	1	1690.10	8.14059e+00	1704.26	1678.04
ft10	1	0609	2	1451.95	6.44351e+00	1460.04	1443.01	ta01	1	0609	2	1846.89	1.09472e+01	1866.81	1824.87
ft10	2	0609	1	1410.66	3.40650e+00	1413.22	1404.69	ta01	2	0609	1	1694.29	1.27254e+01	1719.99	1678.04
ft10	2	0609	2	1453.65	5.79386e+00	1464.30	1443.00	ta01	2	0609	2	1854.23	7.11334e+00	1866.81	1840.59
ft10	6	0609	1	1416.62	3.18629e+00	1421.73	1413.21	ta01	6	0609	1	1711.60	7.11245e+00	1719.99	1699.01
ft10	6	0609	2	1459.19	4.96403e+00	1468.55	1451.52	ta01	6	0609	2	1862.62	8.39200e+00	1882.55	1851.08
ft10	1	0906	1	1418.74	3.32320e+00	1421.73	1413.21	ta01	1	0906	1	1767.71	7.20650e+00	1782.91	1756.70
ft10	1	0906	2	1457.06	5.72775e+00	1464.30	1451.52	ta01	1	0906	2	1888.31	7.58091e+00	1898.28	1872.06
ft10	2	0906	1	1423.00	3.32307e+00	1430.24	1417.47	ta01	2	0906	1	1772.95	7.21050e+00	1788.16	1761.94
ft10	2	0906	2	1457.49	4.74122e+00	1464.30	1451.52	ta01	2	0906	2	1894.61	9.39560e+00	1908.77	1877.30
ft10	6	0906	1	1433.65	3.18335e+00	1438.75	1425.99	ta01	6	0906	1	1799.17	5.95537e+00	1809.13	1788.16
ft10	6	0906	2	1461.74	6.64729e+00	1477.06	1451.53	ta01	6	0906	2	1901.42	7.11201e+00	1908.77	1887.79
ft10	1	0909	1	1403.00	2.08574e+00	1404.70	1400.44	ta01	1	0909	1	1686.43	6.71454e+00	1699.01	1678.04
ft10	1	0909	2	1431.09	5.95943e+00	1438.76	1417.47	ta01	1	0909	2	1808.09	8.71425e+00	1824.87	1793.40
ft10	2	0909	1	1405.97	2.72383e+00	1408.96	1400.45	ta01	2	0909	1	1690.62	4.80603e+00	1699.01	1683.28
ft10	2	0909	2	1437.48	4.68255e+00	1443.01	1430.24	ta01	2	0909	2	1819.62	4.68903e+00	1824.86	1809.13
ft10	6	0909	1	1416.19	3.32551e+00	1421.72	1408.95	ta01	6	0909	1	1729.43	7.34172e+00	1740.97	1719.99
ft10	6	0909	2	1439.60	4.17046e+00	1447.26	1434.49	ta01	6	0909	2	1821.19	7.05490e+00	1835.35	1809.13
ft20	1	0606	1	1648.26	6.70743e+00	1656.88	1637.71	abz7	1	0606	1	974.30	6.04629e+00	982.07	963.64
ft20	1	0606	2	1694.24	7.95341e+00	1704.77	1685.61	abz7	1	0606	2	1054.47	8.02944e+00	1067.64	1039.99
ft20	2	0606	1	1653.05	6.34893e+00	1666.45	1647.30	abz7	2	0606	1	974.04	6.46104e+00	987.33	966.27
ft20	2	0606	2	1695.19	7.41935e+00	1704.77	1685.61	abz7	2	0606	2	1061.58	5.92346e+00	1072.90	1054.47
ft20	6	0606	1	1649.22	5.74367e+00	1656.88	1637.73	abz7	6	0606	1	979.96	9.04772e+00	1000.50	968.90
ft20	6	0606	2	1696.15	5.15936e+00	1704.77	1685.61	abz7	6	0606	2	1061.32	7.13255e+00	1072.90	1050.52
ft20	1	0609	1	1610.91	3.83151e+00	1618.58	1608.99	abz7	1	0609	1	912.16	4.13964e+00	921.51	907.03
ft20	1	0609	2	1662.63	6.35405e+00	1666.46	1647.30	abz7	1	0609	2	1004.18	3.42377e+00	1009.71	999.18
ft20	2	0609	1	1617.62	5.15583e+00	1628.15	1609.00	abz7	2	0609	1	912.95	4.96806e+00	924.14	907.03
ft20	2	0609	2	1666.46	7.41935e+00	1676.04	1656.88	abz7	2	0609	2	1013.00	5.56282e+00	1022.88	1005.76
ft20	6	0609	1	1613.79	4.78900e+00	1618.58	1608.99	abz7	6	0609	1	921.38	3.60196e+00	928.09	913.61
ft20	6	0609	2	1664.55	3.83101e+00	1666.47	1656.88	abz7	6	0609	2	1008.79	3.81714e+00	1013.66	999.18
ft20	1	0906	1	1637.73	7.42064e+00	1647.31	1628.15	abz7	1	0906	1	943.50	5.43132e+00	953.11	932.04
ft20	1	0906	2	1673.16	6.13435e+00	1685.61	1666.45	abz7	1	0906	2	1023.40	5.37118e+00	1032.09	1016.29
ft20	2	0906	1	1641.56	4.68996e+00	1647.31	1637.71	abz7	2	0906	1	956.00	5.45333e+00	963.64	947.84
ft20	2	0906	2	1677.95	7.17115e+00	1695.20	1666.45	abz7	2	0906	2	1036.57	6.03806e+00	1049.21	1026.83
ft20	6	0906	1	1640.60	7.48337e+00	1647.31	1628.14	abz7	6	0906	1	959.56	4.22274e+00	967.59	953.11
ft20	6	0906	2	1675.08	9.03436e+00	1685.62	1656.88	abz7	6	0906	2	1031.04	6.73305e+00	1039.99	1018.93
ft20	1	0909	1	1613.79	4.78800e+00	1618.58	1609.00	abz7	1	0909	1	905.84	2.31278e+00	909.66	903.08
ft20	1	0909	2	1643.48	6.35345e+00	1656.88	1637.72	abz7	1	0909	2	975.49	3.86043e+00	983.38	970.22
ft20	2	0909	1	1615.70	4.38989e+00	1618.58	1608.99	abz7	2	0909	1	904.40	3.99321e+00	913.61	899.13
ft20	2	0909	2	1646.35	5.15806e+00	1656.89	1637.73	abz7	2	0909	2	985.09	4.89118e+00	993.91	976.80
ft20	6	0909	1	1618.57	5.00009e-03	1618.58	1618.57	abz7	6	0909	1	917.30	3.80602e+00	925.46	910.98
ft20	6	0909	2	1644.43	4.38837e+00	1647.31	1637.71	abz7	6	0909	2	979.04	3.72508e+00	987.33	974.17
inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min

A.3.6 Résultats après 300 générations pour 500 individus guidés par f_2

La fonction multicritère f_2 est utilisée pour guider l'AE, les statistiques sur le makespan sont effectivement calculées par l'AE en ce qui concerne la colonne min, mais elles sont estimées après

exécution de l'AE en ce qui concerne la colonne moyenne.

A.3.6.1 Critère $\Pi'_{\min}$

inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	990.40	1.57683e+01	1020	962	ta01	1	0606	1	1344.80	1.36074e+01	1374	1328
ft10	1	0606	2	1000.50	2.84157e+01	1062	967	ta01	1	0606	2	1351.40	1.99710e+01	1385	1323
ft10	2	0606	1	999.60	1.63475e+01	1031	978	ta01	2	0606	1	1373.00	1.16876e+01	1390	1354
ft10	2	0606	2	992.80	1.17371e+01	1015	971	ta01	2	0606	2	1358.90	1.92896e+01	1389	1318
ft10	6	0606	1	991.40	2.35805e+01	1031	963	ta01	6	0606	1	1362.90	2.16308e+01	1389	1314
ft10	6	0606	2	1012.00	2.70592e+01	1055	964	ta01	6	0606	2	1377.60	1.92416e+01	1412	1353
ft10	1	0609	1	1042.00	1.95243e+01	1070	1010	ta01	1	0609	1	1405.70	1.00901e+01	1426	1396
ft10	1	0609	2	978.00	1.66313e+01	1009	945	ta01	1	0609	2	1345.30	7.95047e+00	1359	1332
ft10	2	0609	1	1036.10	2.08636e+01	1067	1005	ta01	2	0609	1	1406.70	1.29696e+01	1423	1380
ft10	2	0609	2	999.80	1.98283e+01	1049	979	ta01	2	0609	2	1355.60	2.28788e+01	1385	1318
ft10	6	0609	1	1061.20	1.79488e+01	1087	1028	ta01	6	0609	1	1419.90	1.15970e+01	1438	1401
ft10	6	0609	2	992.40	1.54480e+01	1008	960	ta01	6	0609	2	1362.90	1.45152e+01	1393	1348
ft10	1	0906	1	974.20	2.01385e+01	1016	934	ta01	1	0906	1	1348.90	1.93000e+01	1382	1318
ft10	1	0906	2	984.80	1.82636e+01	1013	951	ta01	1	0906	2	1333.70	2.71406e+01	1372	1277
ft10	2	0906	1	985.40	1.54350e+01	1011	957	ta01	2	0906	1	1360.30	1.51397e+01	1381	1336
ft10	2	0906	2	1001.60	2.67925e+01	1062	960	ta01	2	0906	2	1357.50	1.56477e+01	1381	1338
ft10	6	0906	1	990.60	1.68297e+01	1020	966	ta01	6	0906	1	1366.20	1.76794e+01	1390	1323
ft10	6	0906	2	991.60	2.35253e+01	1039	959	ta01	6	0906	2	1353.40	1.80677e+01	1388	1332
ft10	1	0909	1	1037.50	1.51212e+01	1062	1017	ta01	1	0909	1	1404.90	8.57263e+00	1421	1389
ft10	1	0909	2	989.30	1.87353e+01	1015	945	ta01	1	0909	2	1342.30	2.05915e+01	1372	1302
ft10	2	0909	1	1036.80	1.68036e+01	1077	1013	ta01	2	0909	1	1413.20	1.27734e+01	1432	1388
ft10	2	0909	2	983.20	1.76737e+01	1013	953	ta01	2	0909	2	1347.70	1.37699e+01	1367	1327
ft10	6	0909	1	1061.60	9.86103e+00	1080	1043	ta01	6	0909	1	1415.80	6.44670e+00	1426	1405
ft10	6	0909	2	984.90	1.64891e+01	1009	960	ta01	6	0909	2	1347.70	1.44503e+01	1367	1317
ft20	1	0606	1	1241.60	1.51275e+01	1270	1218	abz7	1	0606	1	735.00	9.80816e+00	754	721
ft20	1	0606	2	1212.20	1.98635e+01	1254	1183	abz7	1	0606	2	733.10	8.08022e+00	742	716
ft20	2	0606	1	1250.40	2.08096e+01	1283	1221	abz7	2	0606	1	742.70	7.41687e+00	753	729
ft20	2	0606	2	1253.40	1.73505e+01	1295	1228	abz7	2	0606	2	740.00	7.19722e+00	754	728
ft20	6	0606	1	1226.30	1.55824e+01	1254	1199	abz7	6	0606	1	744.60	9.64572e+00	766	732
ft20	6	0606	2	1222.60	2.43113e+01	1260	1186	abz7	6	0606	2	736.70	1.10005e+01	754	723
ft20	1	0609	1	1289.50	2.41340e+01	1332	1253	abz7	1	0609	1	770.40	4.40908e+00	777	761
ft20	1	0609	2	1226.50	1.21511e+01	1246	1210	abz7	1	0609	2	732.70	8.92244e+00	752	723
ft20	2	0609	1	1286.50	2.93061e+01	1329	1244	abz7	2	0609	1	765.80	4.48999e+00	772	756
ft20	2	0609	2	1268.20	1.48916e+01	1289	1238	abz7	2	0609	2	738.60	8.76584e+00	753	726
ft20	6	0609	1	1291.10	1.94497e+01	1336	1263	abz7	6	0609	1	770.10	5.94054e+00	779	758
ft20	6	0609	2	1237.00	2.39040e+01	1276	1200	abz7	6	0609	2	734.60	7.59210e+00	750	722
ft20	1	0906	1	1225.30	1.20669e+01	1249	1208	abz7	1	0906	1	734.60	7.14423e+00	744	721
ft20	1	0906	2	1224.50	2.15743e+01	1249	1192	abz7	1	0906	2	729.70	1.10910e+01	749	711
ft20	2	0906	1	1238.40	1.36909e+01	1260	1218	abz7	2	0906	1	736.10	9.13729e+00	750	723
ft20	2	0906	2	1241.80	1.30062e+01	1256	1212	abz7	2	0906	2	736.90	7.66094e+00	749	726
ft20	6	0906	1	1236.60	1.48203e+01	1254	1213	abz7	6	0906	1	740.70	6.40391e+00	750	731
ft20	6	0906	2	1214.20	1.71511e+01	1249	1184	abz7	6	0906	2	734.90	1.04446e+01	748	712
ft20	1	0909	1	1282.00	1.48862e+01	1305	1259	abz7	1	0909	1	762.80	5.52811e+00	770	754
ft20	1	0909	2	1224.50	1.58193e+01	1248	1198	abz7	1	0909	2	725.30	4.53982e+00	737	720
ft20	2	0909	1	1277.90	1.94137e+01	1325	1260	abz7	2	0909	1	767.50	5.20096e+00	775	756
ft20	2	0909	2	1238.60	2.33589e+01	1290	1205	abz7	2	0909	2	733.20	6.49307e+00	744	722
ft20	6	0909	1	1286.70	1.75901e+01	1331	1268	abz7	6	0909	1	768.80	4.57821e+00	776	760
ft20	6	0909	2	1225.00	2.11092e+01	1262	1188	abz7	6	0909	2	734.90	8.44334e+00	749	723
inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min

A.3.6.2 Critère $\Pi'_{\mathbf{moy}}$

inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	993.09	1.52347e+01	1021.60	962.01
ft10	1	0606	2	1000.32	2.82344e+01	1059.91	966.27
ft10	2	0606	1	999.89	1.60913e+01	1030.12	979.04
ft10	2	0606	2	992.23	1.18048e+01	1013.09	970.52
ft10	6	0606	1	992.23	2.31135e+01	1030.11	962.01
ft10	6	0606	2	1011.81	2.74578e+01	1055.66	962.01
ft10	1	0609	1	1059.91	1.82587e+01	1098.22	1038.63
ft10	1	0609	2	977.76	1.68170e+01	1008.83	944.99
ft10	2	0609	1	1054.81	1.54406e+01	1072.69	1030.12
ft10	2	0609	2	1001.17	1.98560e+01	1047.14	979.04
ft10	6	0609	1	1083.75	1.98931e+01	1106.74	1047.14
ft10	6	0609	2	993.51	1.36188e+01	1008.83	966.27
ft10	1	0906	1	977.33	1.72597e+01	1017.35	949.24
ft10	1	0906	2	985.00	1.80784e+01	1013.09	953.50
ft10	2	0906	1	988.40	1.35672e+01	1013.09	966.27
ft10	2	0906	2	1001.60	2.61029e+01	1059.91	962.01
ft10	6	0906	1	998.62	1.62869e+01	1021.60	970.52
ft10	6	0906	2	991.38	2.32702e+01	1038.63	957.76
ft10	1	0909	1	1072.26	1.35200e+01	1093.96	1047.15
ft10	1	0909	2	989.25	1.81782e+01	1013.09	944.99
ft10	2	0909	1	1065.02	1.23062e+01	1085.45	1042.89
ft10	2	0909	2	982.87	1.75948e+01	1013.09	953.50
ft10	6	0909	1	1107.59	7.32325e+00	1115.25	1089.71
ft10	6	0909	2	987.97	1.65350e+01	1013.09	962.01
ft20	1	0606	1	1243.14	1.64773e+01	1273.79	1216.33
ft20	1	0606	2	1213.46	1.91784e+01	1254.64	1187.60
ft20	2	0606	1	1250.81	2.31441e+01	1283.37	1216.33
ft20	2	0606	2	1253.68	1.68356e+01	1292.95	1225.91
ft20	6	0606	1	1229.74	1.55620e+01	1254.64	1206.75
ft20	6	0606	2	1223.03	2.30834e+01	1264.21	1187.60
ft20	1	0609	1	1315.93	2.19233e+01	1350.41	1283.37
ft20	1	0609	2	1226.86	1.08767e+01	1245.06	1216.33
ft20	2	0609	1	1306.35	2.81512e+01	1331.26	1264.21
ft20	2	0609	2	1269.96	1.36792e+01	1292.95	1245.06
ft20	6	0609	1	1314.02	1.59122e+01	1340.84	1283.37
ft20	6	0609	2	1239.32	2.27468e+01	1273.80	1206.75
ft20	1	0906	1	1234.52	1.31682e+01	1254.64	1206.75
ft20	1	0906	2	1224.95	1.93707e+01	1245.07	1197.17
ft20	2	0906	1	1241.23	1.49603e+01	1264.21	1216.32
ft20	2	0906	2	1243.15	1.27066e+01	1254.64	1216.33
ft20	6	0906	1	1245.06	1.42061e+01	1264.22	1216.33
ft20	6	0906	2	1214.41	1.59095e+01	1245.06	1187.60
ft20	1	0909	1	1316.89	1.43687e+01	1340.84	1292.94
ft20	1	0909	2	1223.99	1.59102e+01	1245.06	1197.17
ft20	2	0909	1	1308.27	1.82711e+01	1350.41	1283.37
ft20	2	0909	2	1239.32	2.35384e+01	1292.95	1206.75
ft20	6	0909	1	1326.47	1.72653e+01	1369.56	1302.52
ft20	6	0909	2	1227.82	2.04518e+01	1264.22	1197.17

inst	op	taux	sch	moy	écart	max	min
ta01	1	0606	1	1352.39	1.63249e+01	1389.63	1337.19
ta01	1	0606	2	1352.39	1.95433e+01	1384.38	1326.70
ta01	2	0606	1	1374.42	1.08606e+01	1389.63	1352.92
ta01	2	0606	2	1359.21	1.98713e+01	1389.62	1316.21
ta01	6	0606	1	1365.51	2.08691e+01	1394.87	1321.46
ta01	6	0606	2	1377.04	1.90804e+01	1410.60	1352.92
ta01	1	0609	1	1423.19	7.48772e+00	1442.06	1415.84
ta01	1	0609	2	1346.10	7.43456e+00	1358.16	1331.94
ta01	2	0609	1	1423.71	1.08101e+01	1436.82	1405.36
ta01	2	0609	2	1356.59	2.20065e+01	1384.38	1316.21
ta01	6	0609	1	1446.78	1.11113e+01	1463.04	1421.09
ta01	6	0609	2	1363.40	1.48337e+01	1394.87	1347.67
ta01	1	0906	1	1356.06	1.95092e+01	1384.38	1321.45
ta01	1	0906	2	1335.09	2.72669e+01	1373.89	1279.51
ta01	2	0906	1	1365.50	1.30992e+01	1384.38	1342.43
ta01	2	0906	2	1358.69	1.54576e+01	1379.14	1337.19
ta01	6	0906	1	1378.09	1.53430e+01	1400.11	1347.67
ta01	6	0906	2	1353.97	1.85848e+01	1389.63	1331.94
ta01	1	0909	1	1433.67	4.80560e+00	1442.06	1426.33
ta01	1	0909	2	1345.05	2.12680e+01	1373.89	1300.48
ta01	2	0909	1	1439.96	8.83678e+00	1452.55	1426.33
ta01	2	0909	2	1350.30	1.39241e+01	1368.65	1326.70
ta01	6	0909	1	1469.86	9.09867e+00	1484.02	1452.55
ta01	6	0909	2	1352.92	1.60776e+01	1368.65	1316.21
abz7	1	0606	1	736.82	9.47795e+00	756.96	725.36
abz7	1	0606	2	733.79	7.68150e+00	742.48	718.78
abz7	2	0606	1	744.05	8.48607e+00	758.27	729.31
abz7	2	0606	2	740.37	7.16736e+00	754.32	727.99
abz7	6	0606	1	747.08	9.29508e+00	767.49	734.58
abz7	6	0606	2	738.00	1.00130e+01	754.32	722.73
abz7	1	0609	1	778.68	3.87136e+00	783.28	772.75
abz7	1	0609	2	733.79	8.83579e+00	753.01	724.04
abz7	2	0609	1	773.15	6.55662e+00	779.34	756.96
abz7	2	0609	2	739.31	8.69687e+00	754.32	727.99
abz7	6	0609	1	781.44	6.20883e+00	789.87	766.17
abz7	6	0609	2	735.89	7.20912e+00	750.37	725.36
abz7	1	0906	1	738.26	5.45415e+00	745.11	727.99
abz7	1	0906	2	729.97	1.10822e+01	749.06	710.88
abz7	2	0906	1	739.05	8.93054e+00	751.69	724.05
abz7	2	0906	2	737.21	7.98642e+00	749.06	725.36
abz7	6	0906	1	748.13	6.00461e+00	758.27	738.53
abz7	6	0906	2	735.24	1.02690e+01	747.74	712.20
abz7	1	0909	1	778.68	1.97434e+00	781.97	775.39
abz7	1	0909	2	726.55	5.28107e+00	741.16	721.41
abz7	2	0909	1	781.18	2.51085e+00	784.60	776.70
abz7	2	0909	2	734.31	7.01088e+00	746.42	721.41
abz7	6	0909	1	788.68	3.69776e+00	793.82	781.97
abz7	6	0909	2	737.60	8.45031e+00	750.37	725.36

A.3.6.3 Critère $\Pi'_{\text{éval}}$

inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	126986.30	1.86302e+02	127251	126586	ta01	1	0606	1	127015.50	2.09982e+02	127549	126759
ft10	1	0606	2	135973.60	3.80161e+02	136446	135232	ta01	1	0606	2	136134.60	3.97461e+02	136992	135627
ft10	2	0606	1	217161.20	3.36128e+02	217623	216453	ta01	2	0606	1	217140.70	3.41560e+02	217607	216544
ft10	2	0606	2	181060.60	2.72501e+02	181510	180624	ta01	2	0606	2	180931.40	2.88755e+02	181350	180604
ft10	6	0606	1	126942.30	1.44653e+02	127233	126754	ta01	6	0606	1	126944.80	1.26010e+02	127133	126648
ft10	6	0606	2	136123.90	3.16849e+02	136726	135614	ta01	6	0606	2	136092.00	3.52869e+02	136670	135533
ft10	1	0609	1	144957.00	1.16429e+02	145095	144707	ta01	1	0609	1	145023.30	1.12934e+02	145168	144825
ft10	1	0609	2	180939.40	2.09531e+02	181354	180535	ta01	1	0609	2	180954.60	2.26110e+02	181254	180549
ft10	2	0609	1	234950.70	1.27312e+02	235106	234709	ta01	2	0609	1	235031.20	1.78501e+02	235286	234715
ft10	2	0609	2	226004.80	3.44462e+02	226530	225426	ta01	2	0609	2	225941.40	2.30601e+02	226496	225706
ft10	6	0609	1	145068.20	7.57401e+01	145173	144927	ta01	6	0609	1	145066.80	5.07263e+01	145154	144991
ft10	6	0609	2	181011.80	1.10847e+02	181146	180823	ta01	6	0609	2	181041.90	1.65281e+02	181377	180746
ft10	1	0906	1	144997.00	4.11582e+01	145061	144932	ta01	1	0906	1	144982.00	8.89078e+01	145151	144823
ft10	1	0906	2	158587.20	2.90197e+02	159174	158144	ta01	1	0906	2	158645.70	2.56212e+02	159196	158320
ft10	2	0906	1	279958.20	1.97055e+02	280254	279601	ta01	2	0906	1	279976.10	1.41567e+02	280235	279761
ft10	2	0906	2	225890.00	2.82854e+02	226358	225330	ta01	2	0906	2	226013.40	2.84869e+02	226634	225588
ft10	6	0906	1	144977.30	7.73900e+01	145101	144846	ta01	6	0906	1	145035.70	4.51044e+01	145122	144965
ft10	6	0906	2	158480.20	3.25701e+02	158989	158061	ta01	6	0906	2	158553.20	2.27101e+02	158930	158173
ft10	1	0909	1	149493.70	2.23609e+01	149538	149451	ta01	1	0909	1	149505.40	3.60144e+01	149577	149448
ft10	1	0909	2	203519.90	1.98597e+02	203763	203124	ta01	1	0909	2	203480.30	1.46079e+02	203718	203304
ft10	2	0909	1	284570.50	1.25072e+02	284760	284431	ta01	2	0909	1	284435.90	1.08669e+02	284694	284308
ft10	2	0909	2	270984.20	2.89136e+02	271434	270582	ta01	2	0909	2	270935.00	1.70871e+02	271202	270602
ft10	6	0909	1	149520.30	2.40751e+01	149571	149490	ta01	6	0909	1	149476.30	1.90108e+01	149518	149454
ft10	6	0909	2	203470.30	1.65571e+02	203668	203153	ta01	6	0909	2	203571.90	2.12405e+02	204023	203218
ft20	1	0606	1	126989.40	1.10182e+02	127182	126772	abz7	1	0606	1	126952.30	1.85062e+02	127221	126577
ft20	1	0606	2	135780.40	2.67886e+02	136243	135308	abz7	1	0606	2	136016.10	2.90616e+02	136519	135460
ft20	2	0606	1	216825.10	3.06225e+02	217196	216183	abz7	2	0606	1	216981.90	2.60261e+02	217530	216585
ft20	2	0606	2	180894.80	2.74988e+02	181260	180200	abz7	2	0606	2	180911.60	2.81544e+02	181344	180448
ft20	6	0606	1	127064.80	1.01357e+02	127205	126879	abz7	6	0606	1	127020.10	2.17988e+02	127480	126761
ft20	6	0606	2	136135.60	3.88719e+02	136630	135292	abz7	6	0606	2	135988.70	3.18437e+02	136428	135399
ft20	1	0609	1	144968.10	1.02982e+02	145157	144850	abz7	1	0609	1	144923.70	5.81808e+01	145038	144856
ft20	1	0609	2	180982.10	2.39434e+02	181423	180706	abz7	1	0609	2	180951.30	2.59756e+02	181354	180455
ft20	2	0609	1	235054.70	1.96056e+02	235426	234814	abz7	2	0609	1	234933.30	1.32871e+02	235088	234676
ft20	2	0609	2	225934.40	2.75184e+02	226434	225354	abz7	2	0609	2	225921.60	2.16820e+02	226248	225610
ft20	6	0609	1	144993.20	8.63444e+01	145104	144857	abz7	6	0609	1	145023.70	7.95375e+01	145147	144852
ft20	6	0609	2	180913.50	1.90622e+02	181335	180627	abz7	6	0609	2	181062.40	1.93490e+02	181313	180730
ft20	1	0906	1	144983.60	4.62195e+01	145055	144901	abz7	1	0906	1	145001.00	3.43831e+01	145061	144943
ft20	1	0906	2	158447.90	3.11474e+02	159010	158007	abz7	1	0906	2	158377.70	3.21008e+02	158867	157741
ft20	2	0906	1	279946.60	8.71920e+01	280091	279790	abz7	2	0906	1	280015.60	1.24088e+02	280216	279800
ft20	2	0906	2	225965.40	2.47977e+02	226294	225618	abz7	2	0906	2	225895.00	2.01953e+02	226132	225428
ft20	6	0906	1	145020.10	8.34715e+01	145163	144913	abz7	6	0906	1	145028.40	8.39192e+01	145192	144885
ft20	6	0906	2	158455.90	2.12539e+02	158968	158196	abz7	6	0906	2	158411.10	2.05281e+02	158846	158096
ft20	1	0909	1	149473.10	5.42945e+01	149540	149388	abz7	1	0909	1	149517.40	2.67365e+01	149556	149472
ft20	1	0909	2	203458.80	1.89638e+02	203797	203150	abz7	1	0909	2	203484.70	2.41236e+02	203893	202899
ft20	2	0909	1	284492.20	1.43750e+02	284676	284255	abz7	2	0909	1	284484.40	1.20691e+02	284747	284329
ft20	2	0909	2	271040.60	2.14719e+02	271546	270754	abz7	2	0909	2	271053.40	2.48826e+02	271534	270566
ft20	6	0909	1	149492.50	4.26433e+01	149552	149411	abz7	6	0909	1	149492.60	2.72991e+01	149540	149460
ft20	6	0909	2	203466.10	1.06717e+02	203584	203219	abz7	6	0909	2	203442.20	2.04101e+02	203723	203093
inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min

A.3.7 Résultats après une génération pour 500 individus guidés par f_3

La fonction multicritère f_3 est effectivement utilisée pour guider l'AE, les statistiques sur le makespan sont calculées directement par l'AE en ce qui concerne la colonne min, mais elles sont

estimées après exécution de l'AE en ce qui concerne la colonne moyenne.

A.3.7.1 Critère $\Pi'_{\min}$

inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	1206.90	3.08300e+01	1248	1151	ta01	1	0606	1	1536.80	3.02880e+01	1578	1497
ft10	1	0606	2	1199.50	2.27079e+01	1240	1159	ta01	1	0606	2	1599.20	3.46635e+01	1633	1526
ft10	2	0606	1	1199.40	2.79005e+01	1252	1152	ta01	2	0606	1	1551.30	1.60751e+01	1579	1524
ft10	2	0606	2	1166.30	2.84009e+01	1241	1136	ta01	2	0606	2	1584.00	2.61151e+01	1633	1528
ft10	6	0606	1	1202.60	3.84999e+01	1243	1129	ta01	6	0606	1	1568.70	9.38136e+00	1589	1552
ft10	6	0606	2	1213.60	3.40065e+01	1274	1146	ta01	6	0606	2	1619.30	2.78031e+01	1642	1561
ft10	1	0609	1	1229.30	3.04633e+01	1271	1173	ta01	1	0609	1	1550.40	1.24676e+01	1566	1532
ft10	1	0609	2	1211.90	2.54615e+01	1250	1156	ta01	1	0609	2	1599.90	2.71089e+01	1642	1555
ft10	2	0609	1	1222.20	2.64568e+01	1272	1175	ta01	2	0609	1	1530.00	2.41288e+01	1572	1496
ft10	2	0609	2	1194.40	3.22341e+01	1251	1151	ta01	2	0609	2	1585.10	3.66427e+01	1636	1494
ft10	6	0609	1	1232.80	1.23353e+01	1251	1215	ta01	6	0609	1	1562.50	1.77553e+01	1584	1531
ft10	6	0609	2	1210.20	1.94000e+01	1251	1177	ta01	6	0609	2	1626.70	1.89528e+01	1642	1579
ft10	1	0906	1	1187.90	2.08012e+01	1221	1157	ta01	1	0906	1	1545.60	7.64461e+00	1563	1535
ft10	1	0906	2	1182.00	2.40333e+01	1219	1132	ta01	1	0906	2	1580.00	1.56652e+01	1621	1558
ft10	2	0906	1	1202.90	2.10829e+01	1245	1166	ta01	2	0906	1	1546.90	1.69378e+01	1572	1521
ft10	2	0906	2	1198.20	2.71912e+01	1240	1149	ta01	2	0906	2	1591.30	1.32819e+01	1618	1572
ft10	6	0906	1	1213.80	2.12923e+01	1250	1167	ta01	6	0906	1	1559.00	1.03730e+01	1574	1537
ft10	6	0906	2	1201.30	3.04008e+01	1238	1141	ta01	6	0906	2	1615.20	2.35449e+01	1642	1574
ft10	1	0909	1	1220.40	2.41917e+01	1253	1177	ta01	1	0909	1	1540.90	2.23358e+01	1562	1483
ft10	1	0909	2	1195.10	1.81684e+01	1231	1171	ta01	1	0909	2	1591.90	2.64894e+01	1622	1525
ft10	2	0909	1	1226.50	3.42615e+01	1302	1184	ta01	2	0909	1	1535.10	1.37437e+01	1550	1514
ft10	2	0909	2	1186.40	3.00440e+01	1217	1117	ta01	2	0909	2	1571.40	3.25859e+01	1610	1504
ft10	6	0909	1	1228.60	2.14485e+01	1258	1186	ta01	6	0909	1	1566.30	1.42622e+01	1583	1536
ft10	6	0909	2	1203.10	2.22821e+01	1235	1160	ta01	6	0909	2	1604.20	2.31249e+01	1642	1559
ft20	1	0606	1	1435.00	3.41789e+01	1490	1377	abz7	1	0606	1	836.20	8.44748e+00	845	815
ft20	1	0606	2	1434.70	1.92200e+01	1464	1404	abz7	1	0606	2	863.20	9.48472e+00	877	846
ft20	2	0606	1	1444.80	2.51110e+01	1504	1414	abz7	2	0606	1	832.90	1.19871e+01	848	814
ft20	2	0606	2	1442.20	2.62861e+01	1486	1398	abz7	2	0606	2	855.30	1.49670e+01	877	827
ft20	6	0606	1	1446.70	1.66075e+01	1467	1419	abz7	6	0606	1	843.50	6.26498e+00	855	832
ft20	6	0606	2	1454.00	1.00000e+01	1471	1432	abz7	6	0606	2	862.30	9.36002e+00	877	848
ft20	1	0609	1	1475.60	2.32086e+01	1522	1423	abz7	1	0609	1	833.80	6.11228e+00	842	822
ft20	1	0609	2	1448.80	2.85615e+01	1474	1375	abz7	1	0609	2	855.80	1.04480e+01	877	839
ft20	2	0609	1	1473.40	3.53870e+01	1523	1416	abz7	2	0609	1	827.80	8.42378e+00	842	809
ft20	2	0609	2	1451.00	2.75536e+01	1498	1397	abz7	2	0609	2	860.20	1.38333e+01	877	832
ft20	6	0609	1	1486.80	2.72500e+01	1530	1440	abz7	6	0609	1	841.10	8.27587e+00	849	820
ft20	6	0609	2	1438.60	3.56180e+01	1495	1385	abz7	6	0609	2	868.80	7.73046e+00	877	857
ft20	1	0906	1	1442.90	1.59465e+01	1474	1418	abz7	1	0906	1	832.60	1.02391e+01	846	814
ft20	1	0906	2	1439.20	1.98383e+01	1469	1401	abz7	1	0906	2	859.20	9.51630e+00	873	838
ft20	2	0906	1	1430.20	1.88722e+01	1463	1401	abz7	2	0906	1	824.00	9.43398e+00	838	808
ft20	2	0906	2	1429.80	2.00290e+01	1468	1399	abz7	2	0906	2	850.70	9.78826e+00	867	836
ft20	6	0906	1	1434.80	2.82588e+01	1481	1381	abz7	6	0906	1	839.10	7.40878e+00	849	822
ft20	6	0906	2	1437.70	3.33708e+01	1472	1360	abz7	6	0906	2	861.10	6.83301e+00	874	848
ft20	1	0909	1	1459.30	2.24234e+01	1498	1433	abz7	1	0909	1	827.40	5.31413e+00	840	819
ft20	1	0909	2	1426.00	3.19124e+01	1473	1364	abz7	1	0909	2	857.20	1.13561e+01	874	841
ft20	2	0909	1	1471.60	2.46382e+01	1505	1415	abz7	2	0909	1	821.50	1.04331e+01	832	801
ft20	2	0909	2	1437.40	1.62247e+01	1463	1412	abz7	2	0909	2	857.40	1.13508e+01	877	832
ft20	6	0909	1	1460.30	2.38833e+01	1490	1417	abz7	6	0909	1	831.90	7.09154e+00	841	815
ft20	6	0909	2	1443.50	1.80569e+01	1484	1411	abz7	6	0909	2	861.20	8.81816e+00	877	850
inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min

A.3.7.2 Critère $\Pi'_{\mathbf{moy}}$

inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	1435.35	6.25423e+00	1447.26	1425.98	ta01	1	0606	1	1797.13	1.53321e+01	1817.87	1768.93
ft10	1	0606	2	1476.21	6.25436e+00	1485.57	1468.55	ta01	1	0606	2	1923.68	1.26014e+01	1950.72	1908.77
ft10	2	0606	1	1435.35	5.31658e+00	1443.01	1425.98	ta01	2	0606	1	1802.03	1.06162e+01	1817.87	1782.92
ft10	2	0606	2	1477.06	4.25600e+00	1481.32	1468.55	ta01	2	0606	2	1931.61	1.48419e+01	1957.71	1911.10
ft10	6	0606	1	1441.31	6.64844e+00	1451.52	1430.24	ta01	6	0606	1	1817.17	9.55639e+00	1834.19	1799.23
ft10	6	0606	2	1482.17	5.95857e+00	1494.09	1472.81	ta01	6	0606	2	1939.06	1.51780e+01	1960.04	1911.09
ft10	1	0609	1	1407.25	2.82185e+00	1413.21	1404.70	ta01	1	0609	1	1693.19	7.31439e+00	1706.01	1678.04
ft10	1	0609	2	1452.80	4.27737e+00	1455.78	1443.01	ta01	1	0609	2	1850.27	5.65644e+00	1859.83	1843.51
ft10	2	0609	1	1408.53	1.27601e+00	1408.96	1404.70	ta01	2	0609	1	1689.22	3.26200e+00	1694.35	1685.03
ft10	2	0609	2	1452.37	4.17067e+00	1460.03	1447.26	ta01	2	0609	2	1854.69	7.27915e+00	1864.48	1841.18
ft10	6	0609	1	1416.62	3.71126e+00	1425.98	1413.21	ta01	6	0609	1	1713.46	8.19512e+00	1726.98	1701.34
ft10	6	0609	2	1453.65	3.43266e+00	1460.03	1447.26	ta01	6	0609	2	1858.42	7.60324e+00	1869.15	1848.17
ft10	1	0906	1	1416.19	3.32755e+00	1421.73	1413.21	ta01	1	0906	1	1768.93	8.00614e+00	1778.25	1750.28
ft10	1	0906	2	1455.78	5.71166e+00	1464.30	1447.27	ta01	1	0906	2	1888.72	1.01669e+01	1901.77	1871.48
ft10	2	0906	1	1421.72	3.80355e+00	1430.23	1417.47	ta01	2	0906	1	1771.96	1.03193e+01	1787.57	1757.28
ft10	2	0906	2	1459.61	4.01644e+00	1464.30	1451.52	ta01	2	0906	2	1898.28	1.07950e+01	1922.75	1883.13
ft10	6	0906	1	1430.66	2.98214e+00	1434.50	1425.98	ta01	6	0906	1	1802.03	1.00379e+01	1815.54	1785.25
ft10	6	0906	2	1461.31	6.03473e+00	1472.80	1451.51	ta01	6	0906	2	1912.03	1.20736e+01	1934.40	1890.12
ft10	1	0909	1	1402.57	3.43204e+00	1408.95	1396.18	ta01	1	0909	1	1686.20	4.06917e+00	1694.35	1680.37
ft10	1	0909	2	1430.24	2.69426e+00	1434.50	1425.98	ta01	1	0909	2	1808.32	5.45936e+00	1815.54	1796.90
ft10	2	0909	1	1406.40	2.08411e+00	1408.96	1404.69	ta01	2	0909	1	1690.62	4.68324e+00	1699.01	1682.70
ft10	2	0909	2	1435.77	4.68191e+00	1443.01	1430.24	ta01	2	0909	2	1818.10	7.32871e+00	1827.20	1806.22
ft10	6	0909	1	1418.32	2.55367e+00	1421.73	1413.21	ta01	6	0909	1	1726.75	9.63223e+00	1743.29	1710.67
ft10	6	0909	2	1442.59	2.29074e+00	1447.27	1438.76	ta01	6	0909	2	1823.70	8.29115e+00	1836.52	1813.21
ft20	1	0606	1	1653.05	6.35345e+00	1666.46	1647.30	abz7	1	0606	1	969.34	8.43193e+00	982.95	956.61
ft20	1	0606	2	1697.11	8.34729e+00	1714.34	1685.61	abz7	1	0606	2	1052.28	6.44774e+00	1061.93	1044.38
ft20	2	0606	1	1654.97	3.83150e+00	1656.89	1647.30	abz7	2	0606	1	979.43	6.44964e+00	991.72	965.39
ft20	2	0606	2	1691.36	6.35586e+00	1704.77	1685.60	abz7	2	0606	2	1061.05	8.28155e+00	1070.71	1048.76
ft20	6	0606	1	1653.05	4.69282e+00	1656.89	1647.29	abz7	6	0606	1	984.26	6.22002e+00	991.72	974.17
ft20	6	0606	2	1698.06	6.12779e+00	1704.76	1685.62	abz7	6	0606	2	1057.98	8.65478e+00	1066.32	1039.99
ft20	1	0609	1	1611.87	4.38946e+00	1618.58	1608.98	abz7	1	0609	1	911.42	2.01154e+00	912.74	908.34
ft20	1	0609	2	1662.63	6.35255e+00	1676.04	1656.87	abz7	1	0609	2	1008.83	3.64565e+00	1013.66	1004.88
ft20	2	0609	1	1611.87	4.39055e+00	1618.58	1608.98	abz7	2	0609	1	912.30	5.35417e+00	921.51	903.96
ft20	2	0609	2	1661.67	6.42273e+00	1676.03	1656.88	abz7	2	0609	2	1011.46	6.28552e+00	1022.44	1004.88
ft20	6	0609	1	1616.66	3.83551e+00	1618.58	1608.98	abz7	6	0609	1	919.32	3.53747e+00	925.90	912.74
ft20	6	0609	2	1663.58	4.39120e+00	1666.46	1656.87	abz7	6	0609	2	1008.39	5.47959e+00	1018.05	1000.50
ft20	1	0906	1	1634.86	6.13419e+00	1647.31	1628.15	abz7	1	0906	1	949.16	4.82746e+00	956.62	943.45
ft20	1	0906	2	1673.16	6.13279e+00	1685.61	1666.45	abz7	1	0906	2	1023.32	5.11649e+00	1031.21	1018.05
ft20	2	0906	1	1640.60	4.39033e+00	1647.31	1637.71	abz7	2	0906	1	953.98	4.88509e+00	961.00	947.84
ft20	2	0906	2	1679.87	6.35224e+00	1685.62	1666.46	abz7	2	0906	2	1031.66	5.35655e+00	1044.38	1022.44
ft20	6	0906	1	1639.64	3.83001e+00	1647.31	1637.72	abz7	6	0906	1	959.25	4.88455e+00	969.78	952.23
ft20	6	0906	2	1677.95	9.38318e+00	1695.19	1666.45	abz7	6	0906	2	1027.70	5.11683e+00	1031.22	1018.05
ft20	1	0909	1	1612.83	4.69200e+00	1618.58	1608.98	abz7	1	0909	1	903.96	2.77648e+00	908.35	899.57
ft20	1	0909	2	1640.60	4.39077e+00	1647.31	1637.72	abz7	1	0909	2	972.41	4.88491e+00	982.94	965.40
ft20	2	0909	1	1617.61	2.87069e+00	1618.58	1609.00	abz7	2	0909	1	907.47	2.63134e+00	912.73	903.96
ft20	2	0909	2	1642.51	4.78701e+00	1647.31	1637.71	abz7	2	0909	2	984.70	4.01957e+00	991.72	978.56
ft20	6	0909	1	1619.53	2.87267e+00	1628.15	1618.56	abz7	6	0909	1	918.44	2.80847e+00	921.51	912.74
ft20	6	0909	2	1644.43	6.13201e+00	1656.88	1637.72	abz7	6	0909	2	979.87	3.42640e+00	982.95	974.17
inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min

A.3.8 Résultats après 300 générations pour 500 individus guidés par f_3

La fonction multicritère f_3 est utilisée pour guider l'AE, les statistiques sur le makespan sont effectivement calculées par l'AE en ce qui concerne la colonne min, mais elles sont estimées après

exécution de l'AE en ce qui concerne la colonne moyenne.

A.3.8.1 Critère $\Pi'_{\min}$

inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	987.60	1.25475e+01	1009	962	ta01	1	0606	1	1354.50	1.18933e+01	1374	1337
ft10	1	0606	2	988.90	9.60677e+00	1007	976	ta01	1	0606	2	1351.60	2.26636e+01	1399	1322
ft10	2	0606	1	998.90	1.92377e+01	1037	976	ta01	2	0606	1	1369.20	1.56767e+01	1397	1349
ft10	2	0606	2	1005.60	1.87521e+01	1045	981	ta01	2	0606	2	1365.90	1.29804e+01	1380	1342
ft10	6	0606	1	996.40	2.30486e+01	1048	971	ta01	6	0606	1	1372.30	1.59440e+01	1388	1343
ft10	6	0606	2	1001.60	2.62343e+01	1047	962	ta01	6	0606	2	1383.90	2.13000e+01	1409	1343
ft10	1	0609	1	1021.80	2.84211e+01	1050	957	ta01	1	0609	1	1404.20	1.30752e+01	1424	1377
ft10	1	0609	2	996.50	2.42291e+01	1047	964	ta01	1	0609	2	1353.90	1.86893e+01	1379	1328
ft10	2	0609	1	1036.80	2.22971e+01	1066	992	ta01	2	0609	1	1407.40	1.18000e+01	1422	1387
ft10	2	0609	2	1007.00	1.75898e+01	1044	984	ta01	2	0609	2	1365.60	1.78617e+01	1386	1340
ft10	6	0609	1	1035.00	1.49733e+01	1058	1017	ta01	6	0609	1	1408.80	1.68452e+01	1434	1380
ft10	6	0609	2	1006.10	1.65436e+01	1023	975	ta01	6	0609	2	1365.70	2.05331e+01	1403	1336
ft10	1	0906	1	972.90	2.09879e+01	1022	951	ta01	1	0906	1	1343.00	1.85849e+01	1363	1306
ft10	1	0906	2	997.40	2.55116e+01	1049	962	ta01	1	0906	2	1341.50	1.21429e+01	1362	1317
ft10	2	0906	1	987.30	1.58496e+01	1024	964	ta01	2	0906	1	1355.20	1.08148e+01	1370	1336
ft10	2	0906	2	996.90	2.48574e+01	1042	960	ta01	2	0906	2	1351.00	2.12273e+01	1400	1316
ft10	6	0906	1	990.10	1.68074e+01	1020	962	ta01	6	0906	1	1372.40	1.05376e+01	1389	1359
ft10	6	0906	2	990.20	1.23839e+01	1015	968	ta01	6	0906	2	1358.70	2.37615e+01	1401	1323
ft10	1	0909	1	1035.80	2.19262e+01	1069	1000	ta01	1	0909	1	1398.50	1.44862e+01	1419	1378
ft10	1	0909	2	980.90	1.51819e+01	1001	954	ta01	1	0909	2	1345.20	1.32650e+01	1372	1324
ft10	2	0909	1	1023.10	2.30150e+01	1049	978	ta01	2	0909	1	1409.00	1.25936e+01	1423	1383
ft10	2	0909	2	986.90	1.66640e+01	1023	965	ta01	2	0909	2	1355.10	1.11126e+01	1374	1342
ft10	6	0909	1	1049.30	1.64441e+01	1077	1019	ta01	6	0909	1	1419.60	1.12089e+01	1432	1403
ft10	6	0909	2	979.00	2.25788e+01	1014	948	ta01	6	0909	2	1356.20	1.79878e+01	1393	1332
ft20	1	0606	1	1228.90	1.63979e+01	1254	1205	abz7	1	0606	1	744.50	6.23298e+00	752	732
ft20	1	0606	2	1218.80	1.39843e+01	1251	1200	abz7	1	0606	2	724.50	6.63702e+00	736	713
ft20	2	0606	1	1256.50	2.42662e+01	1296	1206	abz7	2	0606	1	743.30	6.61891e+00	756	733
ft20	2	0606	2	1246.40	1.81560e+01	1278	1211	abz7	2	0606	2	737.10	7.51598e+00	750	723
ft20	6	0606	1	1239.50	2.18918e+01	1281	1196	abz7	6	0606	1	743.90	9.92421e+00	759	726
ft20	6	0606	2	1239.60	1.40584e+01	1259	1215	abz7	6	0606	2	738.80	9.58958e+00	755	725
ft20	1	0609	1	1291.80	1.85515e+01	1334	1261	abz7	1	0609	1	767.70	4.81768e+00	777	762
ft20	1	0609	2	1223.50	2.20284e+01	1270	1192	abz7	1	0609	2	739.70	1.05456e+01	757	725
ft20	2	0609	1	1291.90	1.32850e+01	1313	1262	abz7	2	0609	1	764.20	8.56505e+00	774	746
ft20	2	0609	2	1253.40	2.48242e+01	1283	1216	abz7	2	0609	2	746.90	1.12201e+01	768	725
ft20	6	0609	1	1292.00	1.40570e+01	1310	1259	abz7	6	0609	1	767.70	8.83233e+00	785	754
ft20	6	0609	2	1233.40	2.30703e+01	1268	1190	abz7	6	0609	2	732.00	8.49706e+00	750	720
ft20	1	0906	1	1224.80	1.36294e+01	1257	1207	abz7	1	0906	1	733.70	5.53263e+00	740	725
ft20	1	0906	2	1228.90	2.21109e+01	1260	1192	abz7	1	0906	2	730.10	1.05019e+01	751	709
ft20	2	0906	1	1246.30	2.03374e+01	1284	1200	abz7	2	0906	1	739.40	7.93977e+00	754	726
ft20	2	0906	2	1242.80	2.10514e+01	1290	1206	abz7	2	0906	2	734.00	9.51840e+00	749	718
ft20	6	0906	1	1223.60	1.11194e+01	1238	1204	abz7	6	0906	1	744.10	8.22739e+00	755	729
ft20	6	0906	2	1223.60	1.67344e+01	1245	1197	abz7	6	0906	2	729.60	1.00717e+01	755	718
ft20	1	0909	1	1280.70	2.59694e+01	1334	1241	abz7	1	0909	1	764.20	6.77938e+00	774	756
ft20	1	0909	2	1212.20	1.08425e+01	1232	1196	abz7	1	0909	2	728.60	6.87314e+00	740	718
ft20	2	0909	1	1282.70	9.39202e+00	1297	1269	abz7	2	0909	1	764.20	7.98499e+00	779	748
ft20	2	0909	2	1239.10	1.51687e+01	1256	1208	abz7	2	0909	2	737.20	5.84466e+00	748	728
ft20	6	0909	1	1288.60	1.30399e+01	1304	1263	abz7	6	0909	1	772.70	7.38986e+00	780	753
ft20	6	0909	2	1221.40	1.56793e+01	1250	1193	abz7	6	0909	2	737.30	7.02922e+00	747	725
inst	op	taux	sch	moy	écart	max	min	inst	op	taux	sch	moy	écart	max	min

A.3.8.2 Critère $\Pi'_{\mathbf{moy}}$

inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	989.25	1.29380e+01	1008.83	962.01
ft10	1	0606	2	988.83	1.00808e+01	1008.83	974.78
ft10	2	0606	1	1000.32	2.14533e+01	1038.63	974.78
ft10	2	0606	2	1006.70	1.81847e+01	1042.89	979.04
ft10	6	0606	1	999.04	2.09428e+01	1047.14	974.78
ft10	6	0606	2	1001.60	2.68566e+01	1047.14	962.01
ft10	1	0609	1	1043.74	2.38352e+01	1072.68	996.07
ft10	1	0609	2	997.34	2.48234e+01	1047.14	962.01
ft10	2	0609	1	1057.79	2.13026e+01	1085.45	1021.60
ft10	2	0609	2	1007.13	1.75724e+01	1042.89	983.29
ft10	6	0609	1	1065.45	1.69248e+01	1085.45	1034.37
ft10	6	0609	2	1006.28	1.57207e+01	1021.60	974.78
ft10	1	0906	1	975.21	1.96396e+01	1021.60	957.75
ft10	1	0906	2	997.34	2.47496e+01	1047.14	962.01
ft10	2	0906	1	987.98	1.59770e+01	1025.86	966.27
ft10	2	0906	2	999.04	2.84300e+01	1047.14	962.01
ft10	6	0906	1	998.19	1.77813e+01	1025.86	970.52
ft10	6	0906	2	990.96	1.36994e+01	1021.60	966.27
ft10	1	0909	1	1070.56	2.50563e+01	1102.48	1025.86
ft10	1	0909	2	981.17	1.40215e+01	1000.32	957.75
ft10	2	0909	1	1055.66	2.43026e+01	1076.94	991.81
ft10	2	0909	2	986.70	1.59033e+01	1021.60	966.27
ft10	6	0909	1	1096.09	1.14619e+01	1110.99	1076.94
ft10	6	0909	2	982.87	2.58532e+01	1030.11	949.24
ft20	1	0606	1	1231.65	1.55626e+01	1254.64	1206.75
ft20	1	0606	2	1218.24	1.53256e+01	1254.64	1197.17
ft20	2	0606	1	1260.38	2.19230e+01	1292.94	1206.75
ft20	2	0606	2	1246.02	1.83983e+01	1273.80	1206.75
ft20	6	0606	1	1243.15	1.99981e+01	1283.38	1206.76
ft20	6	0606	2	1239.31	1.49614e+01	1264.22	1216.33
ft20	1	0609	1	1314.02	1.47171e+01	1340.84	1292.94
ft20	1	0609	2	1225.90	2.26632e+01	1273.79	1197.17
ft20	2	0609	1	1314.02	1.19640e+01	1331.26	1292.94
ft20	2	0609	2	1253.68	2.55000e+01	1283.37	1216.33
ft20	6	0609	1	1317.84	1.66993e+01	1340.83	1283.36
ft20	6	0609	2	1236.44	2.02957e+01	1264.22	1197.17
ft20	1	0906	1	1232.61	1.48657e+01	1264.21	1206.75
ft20	1	0906	2	1228.78	2.26850e+01	1264.21	1187.59
ft20	2	0906	1	1248.89	1.92509e+01	1283.37	1206.75
ft20	2	0906	2	1244.10	2.11801e+01	1292.95	1206.75
ft20	6	0906	1	1231.65	1.36792e+01	1254.63	1216.32
ft20	6	0906	2	1224.95	1.68365e+01	1245.06	1197.17
ft20	1	0909	1	1322.63	1.73729e+01	1350.42	1292.95
ft20	1	0909	2	1215.37	1.38441e+01	1245.06	1197.18
ft20	2	0909	1	1315.93	1.36793e+01	1340.84	1302.52
ft20	2	0909	2	1240.27	1.55908e+01	1264.21	1206.75
ft20	6	0909	1	1325.51	1.06642e+01	1340.83	1312.10
ft20	6	0909	2	1224.95	1.38453e+01	1254.64	1206.75

inst	op	taux	sch	moy	écart	max	min
ta01	1	0606	1	1357.58	1.15723e+01	1379.72	1344.76
ta01	1	0606	2	1351.75	2.26682e+01	1398.37	1321.46
ta01	2	0606	1	1371.80	1.53253e+01	1400.70	1354.09
ta01	2	0606	2	1368.30	1.10533e+01	1379.72	1342.43
ta01	6	0606	1	1376.92	1.41302e+01	1396.03	1351.75
ta01	6	0606	2	1384.62	2.15744e+01	1410.02	1342.43
ta01	1	0609	1	1418.64	1.17014e+01	1435.65	1389.04
ta01	1	0609	2	1354.32	1.86577e+01	1379.72	1328.45
ta01	2	0609	1	1427.96	9.94503e+00	1449.64	1412.35
ta01	2	0609	2	1367.60	1.89560e+01	1389.04	1340.10
ta01	6	0609	1	1438.45	1.49893e+01	1456.63	1414.68
ta01	6	0609	2	1366.20	2.03398e+01	1403.03	1335.44
ta01	1	0906	1	1346.86	1.79466e+01	1370.40	1314.46
ta01	1	0906	2	1342.43	1.20673e+01	1361.08	1316.79
ta01	2	0906	1	1363.17	8.36777e+00	1375.06	1344.76
ta01	2	0906	2	1352.22	2.08160e+01	1400.70	1316.79
ta01	6	0906	1	1385.54	9.67722e+00	1400.69	1370.40
ta01	6	0906	2	1359.21	2.35791e+01	1400.70	1323.79
ta01	1	0909	1	1429.13	5.78525e+00	1442.65	1421.67
ta01	1	0909	2	1347.32	1.32841e+01	1372.73	1326.12
ta01	2	0909	1	1437.52	1.10218e+01	1447.31	1412.35
ta01	2	0909	2	1357.11	1.10342e+01	1375.06	1342.43
ta01	6	0909	1	1467.12	6.18790e+00	1477.60	1456.63
ta01	6	0909	2	1358.28	1.62072e+01	1393.70	1340.10
abz7	1	0606	1	745.98	5.54901e+00	754.76	737.21
abz7	1	0606	2	725.36	7.35451e+00	737.21	715.27
abz7	2	0606	1	745.11	7.28985e+00	754.76	732.82
abz7	2	0606	2	738.09	7.55121e+00	750.37	724.04
abz7	6	0606	1	746.42	9.89871e+00	763.54	728.44
abz7	6	0606	2	739.40	1.00555e+01	754.76	724.04
abz7	1	0609	1	775.83	4.29702e+00	781.09	767.93
abz7	1	0609	2	739.84	1.06051e+01	759.15	724.04
abz7	2	0609	1	771.43	6.74053e+00	776.70	754.76
abz7	2	0609	2	747.74	1.11369e+01	767.93	724.04
abz7	6	0609	1	779.34	6.85369e+00	789.87	767.93
abz7	6	0609	2	734.14	8.33611e+00	750.37	719.66
abz7	1	0906	1	739.40	5.63711e+00	750.37	728.43
abz7	1	0906	2	729.31	1.01587e+01	750.37	710.88
abz7	2	0906	1	742.48	8.04177e+00	754.76	728.44
abz7	2	0906	2	735.46	9.45143e+00	750.37	719.66
abz7	6	0906	1	750.37	7.60024e+00	759.15	732.82
abz7	6	0906	2	730.19	1.07844e+01	759.15	719.66
abz7	1	0909	1	779.34	4.47380e+00	785.48	772.32
abz7	1	0909	2	730.19	7.65285e+00	745.99	719.66
abz7	2	0909	1	778.46	5.26483e+00	785.48	767.93
abz7	2	0909	2	738.09	6.45005e+00	750.37	728.43
abz7	6	0909	1	792.06	2.94267e+00	794.26	785.48
abz7	6	0909	2	738.53	7.61539e+00	745.99	724.04

A.3.8.3 Critère $\Pi'_{\text{éval}}$

inst	op	taux	sch	moy	écart	max	min
ft10	1	0606	1	126910.40	8.41216e+01	127038	126748
ft10	1	0606	2	136055.10	3.19762e+02	136604	135679
ft10	2	0606	1	216996.40	3.51757e+02	217648	216459
ft10	2	0606	2	180954.80	2.28579e+02	181218	180546
ft10	6	0606	1	126940.00	1.77274e+02	127265	126724
ft10	6	0606	2	135940.70	2.61368e+02	136268	135413
ft10	1	0609	1	144979.20	8.43597e+01	145096	144846
ft10	1	0609	2	181034.60	2.44553e+02	181386	180597
ft10	2	0609	1	235094.10	2.48920e+02	235519	234694
ft10	2	0609	2	226028.60	2.39118e+02	226454	225618
ft10	6	0609	1	144981.50	1.16369e+02	145135	144698
ft10	6	0609	2	181196.50	1.36600e+02	181383	180961
ft10	1	0906	1	144997.50	7.57526e+01	145136	144909
ft10	1	0906	2	158533.50	3.54089e+02	159111	158008
ft10	2	0906	1	280042.40	1.35545e+02	280203	279762
ft10	2	0906	2	225870.20	3.62704e+02	226550	225194
ft10	6	0906	1	144999.90	7.24906e+01	145121	144906
ft10	6	0906	2	158490.50	3.31418e+02	159137	157807
ft10	1	0909	1	149495.60	3.38325e+01	149576	149466
ft10	1	0909	2	203515.60	1.50756e+02	203777	203271
ft10	2	0909	1	284542.50	1.26468e+02	284814	284329
ft10	2	0909	2	271023.20	1.49573e+02	271276	270768
ft10	6	0909	1	149501.40	2.51245e+01	149544	149466
ft10	6	0909	2	203520.10	2.21755e+02	203819	203009

inst	op	taux	sch	moy	écart	max	min
ta01	1	0606	1	126982.90	1.21531e+02	127127	126738
ta01	1	0606	2	136084.90	2.90466e+02	136684	135844
ta01	2	0606	1	216944.20	2.93546e+02	217496	216505
ta01	2	0606	2	180843.00	3.57878e+02	181452	180426
ta01	6	0606	1	126935.60	1.46140e+02	127112	126594
ta01	6	0606	2	135961.20	2.07885e+02	136388	135700
ta01	1	0609	1	145006.40	9.53910e+01	145164	144851
ta01	1	0609	2	180997.50	2.04440e+02	181320	180579
ta01	2	0609	1	235002.80	1.93323e+02	235472	234767
ta01	2	0609	2	225963.20	2.58330e+02	226426	225546
ta01	6	0609	1	144983.50	7.73334e+01	145150	144902
ta01	6	0609	2	180934.80	2.50591e+02	181320	180464
ta01	1	0906	1	144999.10	9.37149e+01	145136	144807
ta01	1	0906	2	158599.70	2.20929e+02	158940	158351
ta01	2	0906	1	279952.60	1.30847e+02	280125	279642
ta01	2	0906	2	225942.80	2.90693e+02	226592	225518
ta01	6	0906	1	144978.30	6.29270e+01	145149	144906
ta01	6	0906	2	158464.50	2.03096e+02	158858	158163
ta01	1	0909	1	149476.10	4.29894e+01	149518	149373
ta01	1	0909	2	203446.30	1.58829e+02	203666	203173
ta01	2	0909	1	284495.50	1.72755e+02	284755	284181
ta01	2	0909	2	270998.00	1.70536e+02	271270	270720
ta01	6	0909	1	149527.30	3.48312e+01	149610	149479
ta01	6	0909	2	203544.80	1.71615e+02	203921	203309

inst	op	taux	sch	moy	écart	max	min
ft20	1	0606	1	126989.20	2.40625e+02	127334	126490
ft20	1	0606	2	135929.00	2.20973e+02	136215	135483
ft20	2	0606	1	216989.40	2.99837e+02	217345	216437
ft20	2	0606	2	180968.20	3.09008e+02	181512	180440
ft20	6	0606	1	126855.00	1.45273e+02	127024	126510
ft20	6	0606	2	135747.50	2.66391e+02	136387	135296
ft20	1	0609	1	145016.20	6.20851e+01	145131	144903
ft20	1	0609	2	180896.20	1.78536e+02	181127	180579
ft20	2	0609	1	234899.50	1.72126e+02	235208	234652
ft20	2	0609	2	226101.80	4.83466e+02	227054	225274
ft20	6	0609	1	144978.70	5.51816e+01	145073	144895
ft20	6	0609	2	181037.40	2.10769e+02	181474	180756
ft20	1	0906	1	145043.00	9.24013e+01	145191	144892
ft20	1	0906	2	158481.90	2.37372e+02	158861	158057
ft20	2	0906	1	279947.40	2.01510e+02	280214	279702
ft20	2	0906	2	226124.60	2.61145e+02	226486	225690
ft20	6	0906	1	144996.80	6.11748e+01	145097	144893
ft20	6	0906	2	158555.50	2.51461e+02	158859	158011
ft20	1	0909	1	149532.20	2.69585e+01	149586	149498
ft20	1	0909	2	203529.60	2.33708e+02	203866	203117
ft20	2	0909	1	284533.10	1.02807e+02	284703	284337
ft20	2	0909	2	271071.40	2.61989e+02	271498	270660
ft20	6	0909	1	149520.10	3.37415e+01	149563	149435
ft20	6	0909	2	203485.50	1.05890e+02	203635	203252

inst	op	taux	sch	moy	écart	max	min
abz7	1	0606	1	126994.10	9.64339e+01	127119	126750
abz7	1	0606	2	136089.80	1.86901e+02	136382	135679
abz7	2	0606	1	216941.60	3.12374e+02	217447	216346
abz7	2	0606	2	180899.80	3.67018e+02	181564	180358
abz7	6	0606	1	126943.10	1.93981e+02	127198	126605
abz7	6	0606	2	136115.30	2.23088e+02	136383	135651
abz7	1	0609	1	144977.90	7.11751e+01	145120	144883
abz7	1	0609	2	181040.50	2.33447e+02	181527	180722
abz7	2	0609	1	234875.00	3.11187e+02	235270	234371
abz7	2	0609	2	225933.20	3.35246e+02	226508	225256
abz7	6	0609	1	145019.90	8.01242e+01	145181	144894
abz7	6	0609	2	181002.90	1.82921e+02	181330	180684
abz7	1	0906	1	144954.90	7.20326e+01	145064	144859
abz7	1	0906	2	158534.60	2.85332e+02	158934	158063
abz7	2	0906	1	280027.20	1.15298e+02	280177	279813
abz7	2	0906	2	226088.40	3.15082e+02	226482	225488
abz7	6	0906	1	145014.80	1.12053e+02	145165	144859
abz7	6	0906	2	158439.20	2.37586e+02	158753	158068
abz7	1	0909	1	149471.80	2.95696e+01	149532	149428
abz7	1	0909	2	203464.10	1.65787e+02	203677	203111
abz7	2	0909	1	284489.10	1.16563e+02	284667	284283
abz7	2	0909	2	271106.80	2.52229e+02	271486	270608
abz7	6	0909	1	149511.10	4.49298e+01	149579	149426
abz7	6	0909	2	203469.30	1.05046e+02	203665	203312

A.3.9 Résultats pour ta01 après une génération via l'opérateur invert1

	param		taille de population = 500				taille de population = 900				taille de population = 1600			
	taux	sch	moy	écart	max	min	moy	écart	max	min	moy	écart	max	min
Π'_{min}	0906	1	1550.2	9.46	1567	1537	1535.8	15.86	1556	1507	1521.8	12.31	1540	1504
	0906	2	1583.0	35.82	1627	1487	1583.9	18.93	1610	1547	1558.9	17.50	1585	1532
	0909	1	1543.4	11.19	1563	1524	1535.9	16.50	1556	1510	1521.6	6.62	1534	1511
	0909	2	1608.8	23.97	1637	1567	1557.3	26.83	1588	1498	1562.8	18.33	1601	1539
Π'_{max}	0906	1	2446.4	106.33	2720	2366	2503.1	98.85	2746	2396	2523.7	74.83	2666	2439
	0906	2	2576.1	74.42	2696	2444	2593.9	71.52	2777	2526	2577	64.00	2735	2488
	0909	1	2383.3	86.06	2508	2235	2357.1	85.61	2557	2243	2414.5	66.20	2500	2288
	0909	2	2479.7	66.79	2582	2358	2474.3	44.31	2545	2407	2545.9	23.99	2601	2505
Π'_{moy}	0906	1	1764	6.63	1770	1750	1764	8.00	1770	1750	1763	4.58	1770	1760
	0906	2	1888	9.80	1910	1870	1889	8.31	1900	1880	1886	6.63	1900	1880
	0909	1	1688	4.00	1690	1680	1684	4.90	1690	1680	1685	5.00	1690	1680
	0909	2	1811	7.00	1820	1800	1808	6.00	1820	1800	1809	5.39	1820	1800
$\Pi'_{écart}$	0906	1	173.0	8.21	185	159	172.4	6.53	180	159	171.8	4.21	180	166
	0906	2	229.7	5.59	240	221	229.4	3.69	236	222	228.1	2.12	233	225
	0909	1	112.0	7.68	121	92.9	103.2	7.55	120	93.7	105.5	4.81	116	96.6
	0909	2	202.8	5.83	217	194	198.6	3.90	207	193	201.3	4.96	211	193

A.3.10 Résultats pour ta01 après 1000 générations via l'opérateur invert1

	param		taille de population = 500				taille de population = 900				taille de population = 1600			
	taux	sch	moy	écart	max	min	moy	écart	max	min	moy	écart	max	min
Π'_{min}	0906	1	1349.8	8.85	1365	1336	1315.1	14.18	1335	1290	1316	12.51	1337	1290
	0906	2	1344.3	16.54	1378	1324	1322.2	8.93	1339	1305	1316.3	14.62	1337	1288
	0909	1	1407.9	15.48	1425	1375	1344.7	17.79	1372	1316	1339.2	10.84	1356	1323
	0909	2	1340.3	14.58	1370	1317	1324.1	14.06	1357	1306	1311.6	11.83	1328	1286
Π'_{max}	0906	1	1364.7	9.40	1380	1344	1316.5	13.39	1335	1292	1318.1	11.38	1337	1293
	0906	2	1345.6	16.08	1378	1324	1323.2	9.87	1339	1305	1317.3	15.17	1340	1288
	0909	1	1482.4	10.70	1511	1471	1366.8	12.29	1385	1341	1366.6	17.21	1386	1334
	0909	2	1344.8	12.71	1370	1319	1324.1	14.06	1357	1306	1312.5	11.21	1328	1286
Π'_{moy}	0906	1	1357	7.81	1370	1340	1315	15.00	1340	1290	1317	11.87	1340	1290
	0906	2	1345	16.88	1380	1320	1322	11.66	1340	1300	1317	14.18	1340	1290
	0909	1	1434	10.20	1450	1420	1352	20.40	1380	1320	1347	14.87	1370	1320
	0909	2	1342	13.27	1370	1320	1325	15.00	1360	1310	1313	11.00	1330	1290
$\Pi'_{écart}$	0906	1	2.4	1.47	5.39	0	0.33	0.61	1.76	0	0.23	0.35	0.97	0
	0906	2	0.25	0.44	1.39	0	0.2	0.53	1.79	0	0.24	0.51	1.73	0
	0909	1	12.01	4.03	21.00	7.47	2.95	1.02	5.61	1.89	3.85	2.91	11	0.6
	0909	2	0.87	0.98	3.27	0	0	0	0	0	0.04	0.13	0.42	0
$\Pi'_{éval}$	0906	1	145029	73.6	145137	144913	865817	195.1	866117	865469	1539259	203.6	1539482	1538825
	0906	2	158384	326.6	158975	157681	947081	695.5	948370	946169	1683180	841.9	1685146	1682275
	0909	1	149500	30.4	149542	149453	892811	125.1	892985	892552	1587210	140.0	1587439	1586994
	0909	2	203472	180.8	203751	203191	1216978	341.8	1217700	1216326	2163201	715.7	2164581	2162285

A.3.11 Résultats après une génération pour 1600 individus guidés par f_1, f_2 ou f_3

Lorsque la fonction multicritère f_2 ou f_3 est utilisée pour guider l'AE, les statistiques sur le makespan sont calculées en même temps afin de permettre les comparaisons en terme de makespan. L'opérateur utilisé est invert1, l'instance est ta01.

Les mesures ont été effectuées en activant (avec un taux de 0.3333) et en désactivant la mutation interne de l'opérateur GA/GT.

	param		fonction coût = f_3				fonction coût = f_2				fonction coût = f_1			
	taux	sch	moy	écart	max	min	moy	écart	max	min	moy	écart	max	min
Π'_{min}	0906	1	1524.4	15.69	1556	1495	1523.7	19.45	1543	1485	1521.8	12.31	1540	1504
	0906	2	1567.7	24.98	1612	1523	1571.9	21.86	1609	1531	1558.9	17.50	1585	1532
	0909	1	1530.4	12.01	1549	1513	1527.5	12.77	1543	1504	1521.6	6.62	1534	1511
	0909	2	1570.8	16.32	1590	1539	1566.4	26.72	1610	1504	1562.8	18.33	1601	1539
Π'_{max}	0906	1	2534.6	92.22	2759	2433	2480	65.83	2630	2401	2523.7	74.83	2666	2439
	0906	2	2594.3	47.83	2717	2537	2609.2	72.38	2755	2530	2577	64.00	2735	2488
	0909	1	2400	48.82	2484	2311	2402.1	53.59	2467	2318	2414.5	66.20	2500	2288
	0909	2	2525.2	67.44	2668	2450	2543.3	75.23	2727	2446	2545.9	23.99	2601	2505
Π'_{moy}	0906	1	1763	10.05	1770	1740	1762	6.00	1770	1750	1763	4.58	1770	1760
	0906	2	1885	5.00	1890	1880	1887	6.40	1900	1880	1886	6.63	1900	1880
	0909	1	1685	5.00	1690	1680	1686	4.90	1690	1680	1685	5.00	1690	1680
	0909	2	1810	0	1810	1810	1810	4.47	1820	1800	1809	5.39	1820	1800
$\Pi'_{écart}$	0906	1	169.6	6.90	177	152	167.5	4.18	171	158	171.8	4.21	180	166
	0906	2	228.4	1.50	230	226	229.2	2.44	235	225	228.1	2.12	233	225
	0909	1	105.2	4.59	114	99.7	107.2	4.70	116	99.9	105.5	4.81	116	96.6
	0909	2	200	2.72	207	197	200	2.97	205	194	201.3	4.96	211	193

	param		fonction coût = f_3				fonction coût = f_2				
	taux	sch	moy	écart	max	min	moy	écart	max	min	
Π'_{min}	0906	1	1479.8	16.25	1506	1447	1489.5	10.44	1503	1468	Taux de mutation GA/GT = 0.3333
	0906	2	1494.4	12.75	1524	1475	1499.4	11.94	1522	1480	
	0909	1	1485.7	12.16	1503	1462	1490.2	12.43	1513	1476	
	0909	2	1481.7	18.14	1508	1447	1481.8	19.62	1506	1432	
Π'_{max}	0906	1	2531.9	47.60	2626	2476	2496.4	46.35	2611	2444	
	0906	2	2603.6	44.69	2660	2525	2576.5	57.33	2663	2491	
	0909	1	2430.1	93.90	2629	2306	2441.3	126.1	2752	2324	
	0909	2	2532.1	65.86	2668	2439	2536.8	64.07	2654	2436	
Π'_{moy}	0906	1	1768	6.00	1780	1760	1769	5.39	1780	1760	
	0906	2	1882	4.00	1890	1880	1882	4.00	1890	1880	
	0909	1	1693	4.58	1700	1690	1694	4.90	1700	1690	
	0909	2	1801	3.00	1810	1800	1801	5.39	1810	1790	
$\Pi'_{écart}$	0906	1	171.6	3.23	177	167	171.4	4.03	177	163	
	0906	2	234.9	2.17	239	232	234.7	2.37	239	232	
	0909	1	112.2	6.65	123	101	113.6	5.16	125	107	
	0909	2	206.4	3.93	213	201	207.4	3.61	213	200	

A.3.12 Résultats après 1000 générations pour 1600 individus guidés par f_1, f_2 ou f_3

Lorsque la fonction multicritère f_2 ou f_3 est utilisée, les statistiques sur le makespan sont calculées simultanément par l'AE. L'opérateur utilisé est invert1, l'instance est ta01.

Les mesures ont été effectuées en activant (avec un taux de 0.3333) et en désactivant la mutation interne de l'opérateur GA/GT[1].

[1]Par défaut, lorsque le taux de mutation interne n'est pas précisé, cela signifie que la mutation interne de l'opérateur GA/GT est désactivée).

	param		fonction coût = f_3				fonction coût = f_2				fonction coût = f_1			
	taux	sch	moy	écart	max	min	moy	écart	max	min	moy	écart	max	min
Π'_{min}	0906	1	1309.5	20.09	1342	1270	1304.6	9.33	1316	1287	1316	12.51	1337	1290
	0906	2	1312.4	13.48	1333	1283	1315.4	15.94	1340	1290	1316.3	14.62	1337	1288
	0909	1	1346	15.53	1370	1323	1342.8	13.65	1373	1325	1339.2	10.84	1356	1323
	0909	2	1314.1	12.10	1328	1292	1315.6	15.98	1336	1289	1311.6	11.83	1328	1286
Π'_{max}	0906	1	1313	19.85	1345	1273	1306.5	10.00	1319	1287	1318.1	11.38	1337	1293
	0906	2	1312.4	13.48	1333	1283	1315.4	15.94	1340	1290	1317.3	15.17	1340	1288
	0909	1	1367	11.92	1391	1348	1369.8	12.61	1392	1350	1366.6	17.21	1386	1334
	0909	2	1314.4	12.00	1328	1292	1315.9	16.34	1337	1289	1312.5	11.21	1328	1286
Π'_{moy}	0906	1	1310	20.49	1340	1270	1306	10.20	1320	1290	1317	11.87	1340	1290
	0906	2	1311	13.75	1330	1280	1314	16.25	1340	1290	1317	14.18	1340	1290
	0909	1	1353	11.87	1370	1330	1352	13.27	1380	1330	1347	14.87	1370	1320
	0909	2	1314	13.56	1330	1290	1316	16.25	1340	1290	1313	11.00	1330	1290
$\Pi'_{écart}$	0906	1	0.62	0.64	1.90	0	0.60	1.04	3.31	0	0.23	0.35	0.97	0
	0906	2	0.00	0	0	0	0.00	0	0	0	0.24	0.51	1.73	0
	0909	1	3.49	2.14	7.83	0.2	3.71	1.63	7.19	1.4	3.85	2.91	11	0.6
	0909	2	0.04	0.11	0.37	0	0.12	0.36	1.19	0	0.04	0.13	0.42	0
$\Pi'_{éval}$	0906	1	1539202	280.4	1539792	1538741	1539252	288.1	1539799	1538739	1539259	203.6	1539482	1538825
	0906	2	1682780	845.5	1684118	1681388	1683343	800.4	1685396	1682453	1683180	841.9	1685146	1682275
	0909	1	1587179	172.1	1587419	1586914	1587219	107.9	1587399	1587072	1587210	140.0	1587439	1586994
	0909	2	2163373	486.1	2164488	2162590	2163446	312.0	2164028	2162958	2163201	715.7	2164581	2162285

Taux de mutation GA/GT = 0.3333

	param		fonction coût = f_3				fonction coût = f_2			
	taux	sch	moy	écart	max	min	moy	écart	max	min
Π'_{min}	0906	1	1296.7	12.81	1315	1279	1287.7	11.06	1307	1270
	0906	2	1297.1	19.21	1328	1261	1301.4	18.70	1331	1268
	0909	1	1318.5	12.25	1348	1304	1308.4	9.64	1323	1288
	0909	2	1284.8	19.19	1309	1251	1304.4	16.14	1333	1274
Π'_{max}	0906	1	1297.2	12.39	1315	1279	1288	11.38	1307	1270
	0906	2	1297.1	19.21	1328	1261	1301.4	18.70	1331	1268
	0909	1	1344.3	9.27	1369	1333	1333.1	10.06	1354	1320
	0909	2	1285	18.85	1309	1253	1304.5	16.21	1333	1274
Π'_{moy}	0906	1	1299	13.00	1320	1280	1288	11.66	1310	1270
	0906	2	1298	19.90	1330	1260	1301	19.21	1330	1270
	0909	1	1326	12.81	1360	1310	1319	10.44	1340	1310
	0909	2	1285	19.62	1310	1250	1305	17.46	1330	1270
$\Pi'_{écart}$	0906	1	0.22	0.66	2.19	0	0.05	0.15	0.50	0
	0906	2	0	0	0	0	0	0	0	0
	0909	1	3.83	1.42	6.92	1.86	3.6	1.80	7.69	1.16
	0909	2	0.01	0.04	0.14	0	0.03	0.08	0.28	0
$\Pi'_{éval}$	0906	1	1539255	210.1	1539572	1538963	1539227	171.9	1539515	1538839
	0906	2	1682552	955.2	1684801	1680867	1684094	789.9	1685718	1682651
	0909	1	1587191	110.0	1587416	1587058	1587230	156.2	1587409	1586956
	0909	2	2163424	650.1	2164648	2162747	2163068	622.7	2164182	2162164

A.3.13 Courbes d'évolution

Les deux courbes suivantes présentent l'évolution de la moyenne de Π'_{min} (notée Min sur la courbe) et de la moyenne de Π'_{moy} (notée Moy sur la courbe). Les moyennes sont calculées sur 10 exécutions. Les courbes relatives au schéma séquentiel sont notées Sch1. Les courbes relatives au schéma parallèle sont notées Sch2.

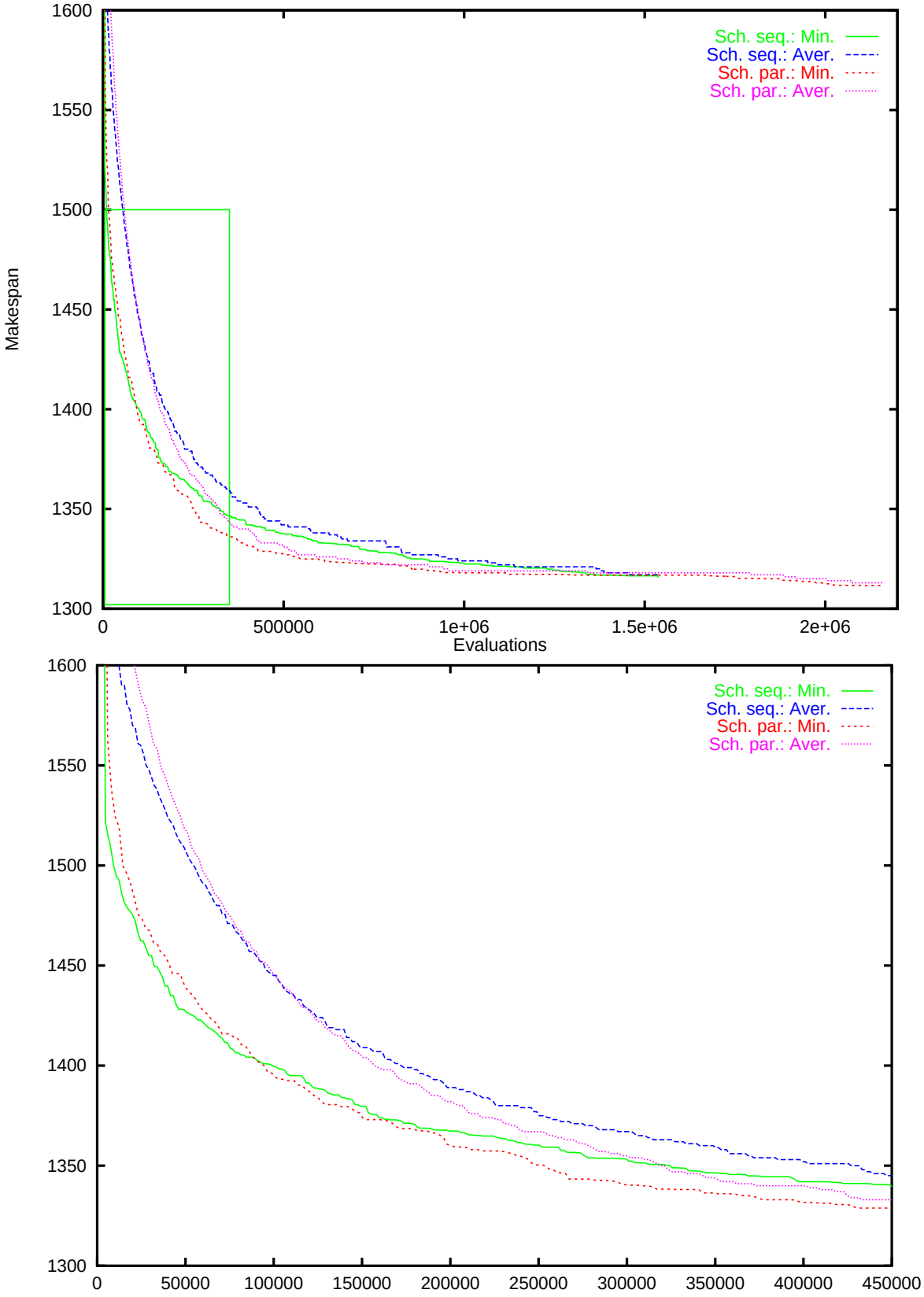
Le paramétrage correspond à celui de la section A.3.12, l'instance étudiée est ta01, la fonction coût est f_1.

Pour chaque schéma, le meilleur paramétrage – en terme de taux d'application des opérateurs – a été retenu parmi les quatres paramétrages utilisés.[2]

Les courbes montrent une convergence plus rapide lors de l'utilisation du schéma parallèle, ce qui pourrait être nuisible après un grand nombre de générations et aboutir à une convergence prématurée. Cependant, comme l'AE est utilisé au sein d'une méthode hybride, le fait d'accélérer la convergence vers de bonnes solutions constitue un avantage non négligeable.

Le gain moyen entre schémas séquentiel et parallèle, mesuré sur les instances étudiées, varie de 0.2% à 6.4%. Il varie en fonction des opérateurs utilisés, des taux d'application des opérateurs, mais aussi en fonction des instances.

[2] le taux de mutation et le taux de crossover sont choisis parmi deux valeurs : 0.6 ou 0.9.



Le second graphe correspond à un agrandissement de la zone encadrée dans le premier. Dans cette zone, les performances du schéma parallèle dépassent les performances du schéma séquentiel (en terme de makespan du meilleur individu et de moyenne de la population).

A.4 Résultats détaillés d'un AE hybride

Un grand nombre d'expériences ont été effectuées pour l'AE et l'AE hybride. Le tableau A.7 synthétise les résultats de plusieurs *benchmarks* obtenus pour différents paramétrages de l'AE-hybride (AE[AIRLNDP]). Un certain nombre de paramètres ne sont pas modifiés pour ces différents *benchmarks* :

- codage : direct ;
- fonction coût : exclusivement basée sur le makespan (f_1) ;
- type d'algorithme évolutif : AG générationnel, élitiste ;
- taux de remplacement : 100% ;
- type de sélection : SUS sans doublon (biais de 1.125) ;
- opérateur de recombinaison : GA/GT (taux de mutation interne fixé à 0.33) ;
- opérateur de mutation : variable, selon expérience ;
- taux de crossover : 0.6 ;
- taux de mutation : ≥ 0.9 variable selon expérience ;
- critère d'arrêt : nombre de générations ;
- taille de population : 300 ;
- nombre de générations : 500 ;
- taux d'application de l'AIRLNDP : 0.05 ;
- opérateur de l'AIRLNDP : variable, selon expérience ;
- schéma d'application des opérateurs : parallèle.

Dans le tableau A.7, la colonne min* donne le meilleur résultat obtenu par l'AE hybride (AE[AIRLNDP]) sur un ensemble de *benchmarks*. Certains résultats correspondent à des *benchmarks* non détaillés dans ce mémoire. Ces résultats n'ont pour but que de valider les opérateurs (et plus généralement les algorithmes) utilisés en montrant que des solutions quasi-optimales peuvent être engendrées pour toutes les instances étudiées.

Instance			Statistiques AE					Statistiques AE[AIRLNDP]					
Inst	JxM	Optimum	exp	moy	écart	max	min	exp	moy	écart	max	min	min*
abz5	10x10	1234	10	1317.30	12.40	1330	1287	10	1249.30	6.94	1261	1238	1234 (0 %)
abz6	10x10	943	10	1015.80	14.86	1035	985	10	955.90	8.63	970	948	943 (0 %)
abz7	20x15	656	15	770.73	5.64	778	757	9	702.56	8.69	715	685	685 (4.42%)
abz8	20x15	645/669	10	812.10	5.37	822	801	10	728.30	8.30	745	716	716 ($\geq$7.03%)
abz9	20x15	661/679	10	814.70	8.19	830	798	10	734.20	9.42	749	722	722 ($\geq$6.33%)
car1	11x5	7038	20	7313.35	119.14	7484	7101	20	7047.05	24.34	7119	7038	7038 (0 %)
car2	13x4	7166	20	7833.95	134.92	8039	7576	10	7349.90	169.85	7617	7166	7166 (0 %)
car3	12x5	7312	20	7642.50	48.89	7757	7594	10	7369.60	38.93	7399	7312	7312 (0 %)
car4	14x4	8003	20	8524.65	68.00	8717	8423	10	8139.60	150.99	8423	8003	8003 (0 %)
car5	10x6	7702	20	8195.30	168.27	8700	8047	10	7755.10	44.65	7821	7702	7702 (0 %)
car6	8x9	8313	10	8772.50	45.03	8871	8699	10	8376.30	72.28	8505	8313	8313 (0 %)
car7	7x7	6558	10	6932.70	103.33	7053	6680	10	6562.50	6.87	6573	6558	6558 (0 %)

Résultats de l'AE et de l'AE hybride .../...

Inst	JxM	Optimum	exp	moy	écart	max	min	exp	moy	écart	max	min	min*
car8	8x8	8264	10	8542.50	62.26	8661	8407	10	8375.10	51.97	8407	8264	8264 (0 %)
la01	10x5	666	10	666.20	0.40	667	666	10	666.00	0.00	666	666	666 (0 %)
la02	10x5	655	10	721.50	6.71	732	711	10	664.00	4.84	669	655	655 (0 %)
la03	10x5	597	10	626.70	0.90	627	624	10	610.50	8.74	620	597	597 (0 %)
la04	10x5	590	10	624.00	6.28	638	620	10	593.40	2.33	598	590	590 (0 %)
la05	10x5	593	10	593.00	0.00	593	593	10	593.00	0.00	593	593	593 (0 %)
la06	15x5	926	10	926.00	0.00	926	926	10	926.00	0.00	926	926	926 (0 %)
la07	15x5	890	10	908.00	9.30	919	892	10	890.00	0.00	890	890	890 (0 %)
la08	15x5	863	10	872.70	7.69	890	863	7	863.00	0.00	863	863	863 (0 %)
la16	10x10	945	10	1025.20	7.93	1038	1016	10	966.30	14.92	982	946	946 (0.11%)
la17	10x10	784	10	839.30	15.24	862	806	10	788.60	3.83	795	784	784 (0 %)
la18	10x10	848	10	890.70	10.97	909	874	10	854.60	5.00	861	848	848 (0 %)
la19	10x10	842	10	895.00	14.12	919	875	10	857.50	8.56	874	847	842 (0 %)
la20	10x10	902	10	929.20	8.34	946	922	10	910.50	1.36	912	907	907 (0.55%)
la21	15x10	1046	10	1163.40	11.59	1186	1143	10	1095.60	10.77	1114	1082	1082 (3.44%)
la22	15x10	927	10	1118.30	20.85	1140	1069	10	969.80	11.98	990	952	952 (2.70%)
la23	15x10	1032	10	1123.10	11.71	1141	1105	10	1038.80	6.88	1052	1032	1032 (0 %)
la24	15x10	935	10	1076.20	14.86	1098	1049	8	972.25	12.52	997	955	955 (2.14%)
la25	15x10	977	10	1139.40	15.84	1163	1120	9	1018.67	18.20	1045	991	991 (1.43%)
la26	20x10	1218	10	1375.40	14.93	1397	1354	9	1243.78	8.93	1257	1232	1232 (1.15%)
la27	20x10	1235	10	1423.20	12.80	1445	1403	9	1302.33	15.33	1325	1276	1276 (3.21%)
la31	30x10	1784	9	1874.00	9.67	1891	1862	10	1784.00	0.00	1784	1784	1784 (0 %)
la32	30x10	1850	10	1937.80	14.10	1956	1912	10	1850.00	0.00	1850	1850	1850 (0 %)
la33	30x10	1719	10	1793.20	15.16	1812	1771	10	1719.00	0.00	1719	1719	1719 (0 %)
la34	30x10	1721	10	1850.70	15.60	1866	1820	7	1728.57	11.47	1756	1721	1721 (0 %)
la35	30x10	1888	10	1980.30	20.12	2009	1943	6	1888.00	0.00	1888	1888	1888 (0 %)
la36	15x15	1268	20	1447.70	16.07	1473	1408	10	1310.60	12.28	1331	1292	1292 (1.89%)
la37	15x15	1397	10	1594.00	8.71	1606	1581	10	1486.60	16.35	1516	1460	1460 (4.51%)
la38	15x15	1196	10	1382.50	10.05	1401	1369	10	1256.70	14.11	1288	1233	1233 (3.09%)
la39	15x15	1233	10	1430.50	10.66	1451	1414	3	1298.67	14.38	1313	1279	1279 (3.73%)
ft6	6x6	55	20	55.00	0.00	55	55	20	55.00	0.00	55	55	55 (0 %)
ft10	10x10	930	10	1064.90	16.94	1083	1038	10	940.10	11.80	967	930	930 (0 %)
ft20	20x5	1165	10	1363.90	12.53	1383	1342	10	1179.30	0.90	1180	1178	1178 (1.12%)
orb1	10x10	1059	10	1208.00	16.08	1221	1165	10	1115.90	18.82	1144	1084	1084 (2.36%)
orb2	10x10	888	10	995.70	11.23	1011	979	8	908.87	9.91	925	894	894 (6.76%)
ta11	20x15	1321/1364	10	1662.90	7.82	1679	1651	3	1493.67	18.66	1520	1479	1479 ($\geq$8.43%)
ta21	20x20	1539/1645	10	2013.40	16.94	2037	1968	10	1798.70	29.80	1838	1721	1721 ($\geq$4.62%)
ta31	30x15	1764	10	2116.30	9.21	2129	2098	10	1927.80	8.74	1942	1916	1907 (8.11%)
ta41	30x20	1859/2023	10	2573.90	12.46	2592	2551	10	2328.90	38.18	2409	2262	2262 ($\geq$11.8%)
ta51	50x15	2760	11	3253.18	18.55	3285	3225	10	3005.90	32.43	3039	2923	2909 (5.40%)
ta71	100x20	5464	1	6065.00	0.00	6065	6065	5	5793.20	52.95	5855	5723	5723 (4.74%)

Tableau A.7: Résultats de l'AE et de l'AE hybride

Annexe B

Comparaison à d'autres auteurs

Le *jobshop* est un problème $\mathcal{NP}$-dur [GS78, GJ79], très difficile à résoudre pratiquement. Pour se convaincre de la difficulté de ce problème, il suffit de mentionner qu'il a fallu attendre plus de 20 ans avant de trouver la solution optimale de certaines instances comportant au plus 100 opérations. Par exemple, l'instance – constituée de dix produits et dix machines – formulée par Fisher et Thompson en 1963 n'a été résolue de manière exacte qu'en 1989 par Carlier et Pinson [CP89]. Pour plusieurs instances de tailles importantes, l'optimum n'a pas encore été obtenu à ce jour.

Le tableau B.1 retrace, dans une certaine mesure, les meilleures performances obtenues par différents auteurs sur les *célèbres* instances de Fisher et Thompson. Nous nous concentrons volontairement sur les publications relatives aux algorithmes évolutifs, antérieures à 1996. Depuis 1996, nous disposons d'un algorithme génétique hybride en mesure de générer des solutions optimales pour ft6 et ft10 (quasi-optimales pour ft20). Cet algorithme nous a permis alors de nous concentrer sur des instances de plus grandes tailles.

Meilleur makespan obtenu (minimum)				
Référence	$\mathcal{A}$	ft6	ft10	ft20
Balas 1969	PSE	**55**	1177	1231
McMahon et Florian 1975	PSE	**55**	972	**1165**
Adams et al. 1988	SBP	**55**	**930**	1178
[CP89]	PSE	**55**	**930**	**1165**
[NY91]	GA	**55**	965	1215
[NY92]	GA	**55**	**930**	1184
[DP92]	GA	**55**	938	1178
[FRC93]	GA		949	1189
[ABN93]	GPp	**55**	943	1182
[JW94]	GA		937	1174
[Soa94]	GA	58	997	
[DPT95c, DPT95a]	HGA	**55**	938	1178
[KOY95]	GA		**930**	1173
[Ghe95]	GA	58	1069	
Optimum		**55**	**930**	**1165**

Tableau B.1: Comparaison à d'autres auteurs. La colonne « $\mathcal{A}$ » précise l'algorithme utilisé en utilisant le code suivant : GA - algorithme génétique ; GPp - programmation génétique parallèle ; PSE - procédure par séparation et évaluation ; SBP : *Shifting Bottleneck Procedure* ; HGA : algorithme génétique hybride

Annexe C

Résultats de la OR-Library

La *OR-Library* [Bea90] est une bibliothèque constituée d'un très grand nombre d'instances de différents problèmes largement utilisés dans la « communauté recherche opérationnelle ».

Ces instances sont disponibles sur le site ftp `ftp://mscmga.ms.ic.ac.uk/pub` ou sur le site WWW `http://mscmga.ms.ic.ac.uk/info.html`.

Nous ne présentons ici que les instances de *jobshop* simple. Ces instances sont réparties en deux catégories :

- la première est constituée d'instances proposées par différents auteurs à partir de problèmes plus ou moins réels ;
- la seconde est constituée d'instances générées par le générateur d'instances proposé par Éric Taillard.

Pour chaque instance, les méthodes ayant donné les meilleurs résultats, ou une solution optimale pour la première fois, sont indiquées. En conséquence, la *OR-Library* constitue une source d'informations très utiles lorsqu'il s'agit de comparer des méthodes entre-elles.

C.1 Première partie

Cette partie est constituée de 82 instances de *jobshop* (dernière mise à jour 23/02/96) rassemblées par Dirk C. Mattfeld (email dirk@uni-bremen.de) et Rob J.M. Vaessens (email robv@win.tue.nl) :

Instances	Instances issues de
abz5-abz9	J. Adams, E. Balas and D. Zawack (1988), The shifting bottleneck procedure for jobshop scheduling, Management Science 34, 391-401.
ft06, ft10, ft20	H. Fisher, G.L. Thompson (1963), Probabilistic learning combinations of local jobshop scheduling rules, J.F. Muth, G.L. Thompson (eds.), Industrial Scheduling, Prentice Hall, Englewood Cliffs, New Jersey, 225-251.
la01-la40	S. Lawrence (1984), Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement), Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania.
orb01-orb10	D. Applegate, W. Cook (1991), A computational study of the jobshop scheduling instance, ORSA Journal on Computing 3, 149-156. (they were generated in Bonn in 1986).
swv01-swv20	R.H. Storer, S.D. Wu, R. Vaccari (1992), New search spaces for sequencing instances with application to jobshop scheduling, Management Science 38, 1495-1509.
yn1-yn4	T. Yamada, R. Nakano (1992), A genetic algorithm applicable to large-scale jobshop instances, R. Manner, B. Manderick (eds.), Parallel instance solving from nature 2, North-Holland, Amsterdam, 281-290.

Pour chaque instance, les tableaux suivants donnent la valeur optimale (si elle est connue), ou les meilleures bornes inférieures et supérieures connues à ce jour. La taille de chaque instance est également précisée, ainsi qu'un code référençant le premier auteur (ou groupe d'auteurs) qui ont prouvé la valeur de la borne inférieure, ou trouvé une solution dont la valeur est la borne supérieure. La liste des codes utilisés est située sous les tableaux.

abz5-6 10 x 10	1234 AC AC	943 AC ABZ			
abz7-9 20 x 15	656 MS MS	645- 669 MS BV	661- 679 MS BV		
ft06 6 x 6	55 LLR LLR				
ft10 10 x 10	930 CP1 La				
ft20 20 x 5	1165 CP1 CP1				
la01-05 10 x 5	666 ABZ ABZ	655 ABZ LAL	597 AC MSS	590 AC LAL	593 ABZ ABZ
la06-10 15 x 5	926 ABZ ABZ	890 ABZ ABZ	863 ABZ ABZ	951 ABZ ABZ	958 ABZ LAL
la11-15 20 x 5	1222 ABZ ABZ	1039 ABZ ABZ	1150 ABZ ABZ	1292 ABZ ABZ	1207 ABZ ABZ
la16-20 10 x 10	945 CP2 CP2	784 CP2 MSS	848 AC MSS	842 AC MSS	902 AC LAL
la21-25 15 x 10	1046 VAL VAL	927 AC MSS	1032 ABZ ABZ	935 AC AC	977 AC AC
la26-30 20 x 10	1218 ABZ LAL	1235 ABZ CP3	1216 ABZ MSS	1142-1153 NR VAYN	1355 ABZ ABZ
la31-35 30 x 10	1784 ABZ ABZ	1850 ABZ ABZ	1719 ABZ ABZ	1721 ABZ ABZ	1888 ABZ ABZ
la36-40 15 x 15	1268 CP2 CP2	1397 CP2 AC	1196 VAL NS	1233 AC AC	1222 AC AC
orb01-05 10 x 10	1059 AC AC	888 AC AC	1005 AC AC	1005 AC AC	887 AC AC
orb06-10 10 x 10	1010 AC AC	397 AC AC	899 AC AC	934 AC AC	944 AC AC
swv01-05 20 x 10	1392-1418 VA BV	1475 VA VABV	1369-1398 VA VAWe	1450-1483 VA BV	1421-1434 VA BV
swv06-10 20 x 15	1591-1696 VA BV	1446-1620 VA VABV	1640-1770 VA VAAa	1604-1663 VA VABV	1631-1773 VA BV
swv11-15 50 x 10	2983-3005 VA VABV	2972-3038 VA BV	3104-3146 VA Aa	2968 VA BV	2885-2940 VA VAAa
swv16-20 50 x 10	2924 VA SWV	2794 VA SWV	2852 VA SWV	2843 VA SWV	2823 VA SWV
yn1-4 20 x 20	826- 888 CL We	861- 909 CL VABV	827- 894 VA VABV	918- 972 VA BV	

Aa	B. Aarts (1996), Personal communication. (a parallel taboo search algorithm).
ABZ	J. Adams, E. Balas and D. Zawack (1988), The shifting bottleneck procedure for jobshop scheduling. Management Sci. 34, 391-401.
AC	D. Applegate, W. Cook (1991), A computational study of the jobshop scheduling problem. ORSA J. Comput. 3, 149-156.
BV	E. Balas, A. Vazacopoulos (1995), Guided local search with shifting bottleneck for jobshop scheduling. Management Science Research Report #MSRR-609(R) [revised version], Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania.
CL	Y. Caseau, F. Laburthe (1995), Disjunctive scheduling with task intervals. Report LIENS 95-25, Département de Mathématiques et Informatique, École Normale Supérieure, Paris, France.
CP1	J. Carlier, E. Pinson (1989), An algorithm for solving the jobshop problem. Management Science 35, 164-176.
CP2	J. Carlier, E. Pinson (1990), A practical use of Jackson's preemptive schedule for solving the jobshop problem. Ann. Oper. Res. 26, 269-287.
CP3	J. Carlier, E. Pinson (1994), Adjustments of heads and tas for the jobshop problem. European J. Oper. Res. 78, 146-161.
La	B.J. Lageweg (1984), Personal communication.
LAL	P.J.M. Van Laarhoven, E.H.L. Aarts, J.K. Lenstra (1992), Jobshop scheduling by simulated annealing. Oper. Res. 40, 113-125.
LLR	B.J. Lageweg, J.K. Lenstra, A.H.G Rinnooy Kan (1977), jobshop scheduling by implicit enumeration. Management Science 24, 441-450.
MS	P. Martin, D.B. Shmoys (1995), A new approach to computing optimal schedules for the jobshop scheduling problem. Working paper, School of Operations Research an Industrial Engineering, Cornell University, Ithaca, New York.
MSS	H. Matsuo, C.J. Suh, R.S. Sullivan (1988), A controlled search simulated annealing method for the general jobshop scheduling problem. Working paper 03-04-88, Graduate School of Business, University of Texas, Austin.
NR	W. Nuijten, J. Rogerie (1996), Communicated by post on netnews.sci.op-research on Mon, 22 Jan 96.
NS	E. Nowicki, C. Smutnicki (1993), A fast taboo search algorithm for the jobshop problem. Preprinty nr 8/93, Institute of Engineering Cybernetics, Technical University of Wroclaw, Wroclaw, Poland.
SWV	R.H. Storer, S.D. Wu, R. Vaccari (1992), New search spaces for sequencing problems with application to jobshop scheduling. Management Sci. 38, 1495-1509.
VA	R.J.M. Vaessens (1995), Personal communication. (lower bounds found using Applegate & Cook's algorithm "edge-finder").
VAAa	R.J.M. Vaessens (1996), Personal communication. (upper bounds found using Applegate & Cook's algorithm "shuffle" with initial solutions of B. Aarts (1996)).
VABV	R.J.M. Vaessens (1995), Personal communication. (upper bounds found using Applegate & Cook's algorithm "shuffle" with initial solutions of E. Balas, A. Vazacopoulos (1994)).
VABV	R.J.M. Vaessens (1996), Personal communication. (upper bounds found using Applegate & Cook's algorithm "shuffle" with initial solutions of E. Balas, A. Vazacopoulos (1996)).
VAL	R.J.M. Vaessens, E.H.L. Aarts, J.K. Lenstra (1995), Jobshop scheduling by local search. Memorandum COSOR 94-05 (second revised version) Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, The Netherlands.
VAWe	R.J.M. Vaessens (1995), Personal communication. (upper bounds found using Applegate & Cook's algorithm "shuffle" with initial solutions of M. Wennink (1995)).
VAYN	R.J.M. Vaessens (1995), Personal communication. (upper bounds found using Applegate & Cook's algorithm "shuffle" with initial solutions of T. Yamada, R. Nakano (1995)).
We	M. Wennink (1995), Personal communication. (upper bounds found by a taboo search algorithm).
YN	T. Yamada, R. Nakano (1995), Jobshop scheduling by simulated annealing combined with deterministic local search, MIC-95.

C.2 Deuxième partie

Cette partie est constituée de 80 instances de *jobshop* générées par le générateur d'instances proposé par Éric Taillard [Tai93a] (dernière mise à jour 30/06/99 sur la *OR-Library*, et 27/01/99 sur http://www.idsia.sh/~eric/problemes.dir/ordonnancement.dir/jobshop.dir/best_lb_up.txt).

Pour chaque instance, les tableaux suivants donnent la valeur optimale (si elle est connue), ou les meilleures bornes inférieures et supérieures connues à ce jour. La taille de chaque instance est également précisée, ainsi qu'un code référençant le premier auteur (ou groupe d'auteurs) qui ont prouvé la valeur de la borne inférieure, ou trouvé une solution dont la valeur est la borne supérieure. La liste des codes utilisés est située sous les tableaux.

ta01-05	1231	1244	1218	1175	1224
15 x 15	Ta1 Ta1	VA NS	B BV	B We	B B
ta06-10	1238	1227	1217	1274	1241
15 x 15	B B	B B	B BV	B BV	VA BV
ta11-15	1321-1364	1321-1367	1271-1350	1345	1293-1342
20 x 15	VA BV	VA BV	VA BV	VA NS	VA VAAa
ta16-20	1300-1362	1458-1464	1369-1396	1276-1341	1316-1353
20 x 15	VA AHLS	VA BV'	VA BV	VA BV	VA We
ta21-25	1539-1645	1511-1601	1472-1558	1602-1651	1504-1597
20 x 20	VA AHLS	VA Aa	VA BV	VA Aa	VA AHLS
ta26-30	1539-1651	1616-1687	1591-1615	1514-1625	1473-1585
20 x 20	VA AHLS	VA AHLS	VA BV	VA Aa	VA AHLS
ta31-35	1764	1774-1803	1778-1796	1828-1832	2007
30 x 15	Ta1 AHLS	Ta1 BV	VA BV	Ta1 BV	VA Ta1
ta36-40	1819	1771-1784	1673-1677	1795	1631-1686
30 x 15	VA AHLS	Ta1 BV	Ta1 AHLS	VA AHLS	VA BV
ta41-45	1859-2023	1867-1961	1809-1879	1927-1998	1997-2005
30 x 20	VA BV'	VA BV	VA BV	VA AHLS	VA BV
ta46-50	1940-2029	1789-1913	1912-1971	1915-1984	1807-1937
30 x 20	Ta1 AHLS	VA AHLS	VA AHLS	VA AHLS	VA AHLS
ta51-55	2760	2756	2717	2839	2679
50 x 15	Ta1 Ta1	Ta1 Ta1	Ta1 Ta1	Ta1 Ta1	Ta1 NS
ta56-60	2781	2943	2885	2655	2723
50 x 15	Ta1 Ta1	Ta1 Ta1	Ta1 Ta1	Ta1 Ta1	Ta1 Ta1
ta61-65	2868	2869-2872	2755	2702	2725
50 x 20	Ta1 NS	VA AHLS	Ta1 NS	BV NS	Ta1 NS
ta66-70	2845	2825	2784	3071	2995
50 x 20	Ta1 NS	VA AHLS	BV NS	Ta1 NS	Ta1 NS
ta71-75	5464	5181	5568	5339	5392
100 x 20	Ta1 Ta1	Ta1 Ta1	Ta1 Ta1	Ta1 Ta1	Ta1 Ta1
ta76-80	5342	5436	5394	5358	5183
100 x 20	Ta1 Ta1	Ta1 Ta1	Ta1 Ta1	Ta1 Ta1	Ta1 NS

Aa	B. Aarts (1996), Personal communication. (a parallel taboo search algorithm)
AHLS	E. Aarts, Huub ten Eikelder, J.K. Lenstra, R. Schilham (23.6.1999), Personal communication. (An adaptive memory programm embedding a taboo search algorithm with NS neighbourhood and constant time Ta)
B	Wolfgang Brinkkoetter (27.1.1999), Personal communication. (The method combines ideas of Carlier and Pinson with an efficient way of propagating release/queue-time adjustments in a simulation of parallel preemptive machines.)
BV	E. Balas, A. Vazacopoulos (1995), Guided local search with shifting bottleneck for jobshop scheduling. Management Science Research Report #MSRR-609(R) [revised version], Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania.
BV'	E. Balas, A. Vazacopoulos (1996), Personal communication.
NS	E. Nowicki, C. Smutnicki (1993), A fast taboo search algorithm for the jobshop problem. Preprinty nr 8/93, Institute of Engineering Cybernetics, Technical University of Wroclaw, Wroclaw, Poland.
Ta1	E. Taillard (1993), Benchmarks for basic scheduling problems. EJOR 64, 278-285.
VA	R.J.M. Vaessens (1995), Personal communication. (lower bounds found using Applegate & Cook's algorithm "edge-finder").
VAAa	R.J.M. Vaessens (1996), Personal communication. (upper bounds found using Applegate & Cook's algorithm "shuffle" with initial solutions of B. Aarts (1996)).
VABV	R.J.M. Vaessens (1996), Personal communication. (upper bounds found using Applegate & Cook's algorithm "shuffle" with initial solutions of E. Balas and A. Vazacopoulos (1995)).
VAWe	R.J.M. Vaessens (1995), Personal communication. (upper bounds found using Applegate & Cook's algorithm "shuffle" with initial solutions of M. Wennink (1995)).
We	M. Wennink (1995), Personal communication. (upper bounds found by a taboo search algorithm).

Annexe D

Outils utilisés

Pour mener à bien nos travaux nous avons réalisé ou utilisé un certain nombre d'outils que nous présentons dans cette annexe.

D.1 LibGA

La version 1.0 de *LibGA*, réalisée en 1993 par Arthur L. Corcoran est disponible sur de nombreux sites FTP. Il s'agit d'une bibliothèque d'algorithmes génétiques séquentiels écrite en langage C utilisable sous MSDOS comme sous UNIX. Elle comporte les principaux opérateurs génétiques standards ainsi que deux moteurs d'algorithmes génétiques génériques : générationnel et stationnaire. Organisé à la manière d'une bibliothèque d'opérateurs, *LibGA* est facilement extensible et applicable à de nombreux problèmes.

Nous avons poursuivi le développement de cette librairie. La version actuelle est écrite en C/Ansi et fonctionne uniquement sous Unix. elle a été évaluée sur différents environnements parmi lesquels : Solaris, Linux, Irix, Osf. Elle possède à ce jour les caractéristiques suivantes :

- algorithme génétique générationnel ou stationnaire ;
- chromosomes de type tableaux d'entiers ou de nombres réels, permutations d'entiers ;
- accepte une taille de population variable ;
- opérateurs de recombinaison simples (un point, uniforme) ou dédiés au TSP (pmx, er, uox, rox, ...) ;
- opérateurs de mutation simples ou dédiés au TSP (2-opt, ...) ;
- opérateurs de sélection de type roulette, SUS, ...

Il est possible de manipuler des chromosomes codés sous forme de chaînes de bits ou de caractères au prix d'un coût mémoire important[1].

Actuellement *LibGA* n'est plus mise à jour par Arthur L. Corcoran. Cependant nous continuons à faire évoluer une version séquentielle de LibGA au sein du LIL. Nous avons utilisé cette version de *LibGA* pour réaliser nos algorithmes évolutifs hybrides appliqués au *jobshop*.

De plus, dans le cadre du groupe PERFORM une version parallèle de *LibGA* basée sur l'environnement de programmation parallèle *MARS* [HTG97, HTG96] a vu le jour : *LibGA_p*.

[1] Chaque gène est vu comme un *double* (nombre codé en virgule flottante, double précision)

D.2 Visusch

Visusch est un outil graphique conçu dans le cadre de nos travaux afin d'étudier les différents critères utilisables dans les fonctions coût et mettre au point les opérateurs. Il permet de visualiser les ordonnancements et d'appliquer un ensemble de traitements sur les opérations (marquages, déplacement, opérateurs de mutation, opérateurs de recombinaison). Il est également utilisable pour évaluer un critère particulier sur toutes les instances de la *OR-Library* (voir tableau A.6, page 180).

Visusch fonctionne sur des stations Silicon Graphics sous Irix et sur IBMPC sous Linux. La figure D.1 présente un exemple d'utilisation de Visusch.

Figure D.1: Exemple d'utilisation de Visusch.
Note : les couleurs ont été modifiées par rapport à l'affichage initial de manière à faciliter l'impression.

Bibliographie

[AA92] D. Abramson et J. Abela. A parallel genetic algorithm for solving the school timetabling problem. In *15th Australian Computer Science Conference*, Février 1992.

[AAK+93] J. Abela, D. Abramson, M. Krishnamoorthy, A. De Silva, et G. Mills. Computing optimal schedules for landing aircraft, Mai 1993.

[ABN93] Laurent Atlan, Jérôme Bonnet, et Martine Naillon. Learning distributed reactive strategies by genetic programming for the general job shop scheduling. In *Proceedings of the* 7^{th} *Florida Artificial Intelligence Research Symposium (FLAIRS)*, 1993.

[AC94] Marco Adinolfi et Amedeo Cesta. Scheduling heuristics for the rds-sched system. In *[Eib94]*, pages 13–17, 1994.

[Ack87] David H. Ackley. An empirical study of bit vector function optimization. In *[Dav87]*, pages 170–204. Morgan Kaufmann, San Mateo, CA, USA, 1987.

[AD] D. Abramson et H. Dang. School timetables: A case study in simulated annealing.

[Ada94] Panagiotis Adamidis. Review of parallel genetic algorithms bibliography. Rapport technique Version 1, Aristotle University of Thessaloniki, Faculty of Engineering, Department of Electrical And Computer Engineering, Division of Electronics and Computers, Thessaloniki, Greece, Novembre 1994.

[AF90] A. Alj et R. Faure. *Guide de la recherche opérationnelle, tome 2.* Masson, 1990. ISBN: 2-903-60761-3.

[AFH95] Aziz El Abdellaoui, A. Fréville, et S. Hanafi. Métaheuristiques surrogates pour le sac-à-dos bidimensionnel en variables 0-1. In *Francoro 95*, page 26, Juin 1995. (one page summary).

[Aga98] Alexandru Agapie. Modelling genetic algorithms : From markov chains to dependence with complete connections. In *[EBSS98]*, pages 3–12, Septembre 1998.

[ALRS94] J-M. Alliot, E. Lutton, E. Ronald, et M. Schoenauer, éditeurs. *Actes de Évolution Artificielle'94*, Toulouse, France, Septembre 1994. Cépaduès. ISBN: 2-85428-411-9.

[AM95] Tolga Asveren et Paul Molitor. New crossover methods for sequencing problems. In *[Esh95]*, pages 290–299, 1995.

[AR96] D. Anciaux et D. Roy. Reactive production system: A new approach. In *[FUC96]*, pages 277–281, Septembre 1996.

[AS94] Robert J. Aarts et Stephen F. Smith. A high performance scheduler for an automated chemistry workstation. In *[Eib94]*, pages 3–7, 1994.

[Bac95] Vincent Bachelet. Heuristiques parallèles hybrides appliquées à l'affectation quadratique. Master's thesis, Université des Sciences et Technologies de Lille 1, Villeneuve d'Ascq, Juillet 1995. DEA thesis, Laboratoire d'Informatique Fondamentale de Lille.

[Bac99] Vincent Bachelet. *Métaheuristiques parallèles hybrides : application au problème d'affectation quadratique.* PhD thesis, Université des Sciences et Technologies de Lille I, Lille, Décembre 1999.

[Bak85] James Edward Baker. Adaptive selection methods for genetic algorithms. In *[Gre85]*, pages 101–111, 1985.

[Bak87] James Edward Baker. Reducing bias and inefficiency in the selection algorithm. In *[Gre87a]*, pages 14–21, 1987.

[BATM93] Pierre Bessière, Juan-Manuel Ahuactzin, El-Ghazali Talbi, et Emmanuel Mazer. The ariadne's clew algorithm: Global planning with local methods. In *Intelligent Robots and Systems*. IEEE, 1993.

[BB91] Richard K. Belew et Lashon B. Booker, éditeurs. *Proceedings of the Fourth International Conference on Genetic Algorithms*, La Jolla, CA, USA, Juillet 1991. Morgan Kaufmann, San Mateo, CA, USA. ISBN: 1-55860-208-9.

[BBM93a] David Beasley, David R. Bull, et Ralph R. Martin. An overview of genetic algorithms: Part 1, fundamentals. *University Computing*, 15(2):58–69, 1993.

[BBM93b] David Beasley, David R. Bull, et Ralph R. Martin. An overview of genetic algorithms: Part 2, research topics. *University Computing*, 15(4):170–181, 1993.

[BBM93c] David Beasley, David R. Bull, et Ralph R. Martin. Reducing epistasis in combinatorial problems by expansive coding. In *[For93]*, pages 400–407, 1993.

[BBM93d] David Beasley, David R. Bull, et Ralph R. Martin. A sequential niche technique for multimodal function optimization. *Evolutionary Computation*, 1(2):101–125, 1993. ISSN: 1063-6560.

[Bäc92] Thomas Bäck. The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. In *[MM92]*, 1992.

[Bäc94] Thomas Bäck. Parallel optimization of evolutionary algorithms. In *[DSM94]*, pages 418–427, 1994.

[Bäc97] Thomas Bäck, éditeur. *Proceedings of the Seventh International Conference on Genetic Algorithms*, East Lansing, Michigan, USA, Juillet 1997. Morgan Kaufmann, San Mateo, CA, USA. ISBN: 1-558-60487-1.

[BCL+93] F. Baiardi, D. Ciuffolini, A.M. Lomartire, D. Montanari, et G. Pesce. Nested hybrid genetic algorithms for system configuration and program mapping in massively parallel systems. In *[For93]*, page 626, 1993. (one page summary).

[BD93] Alberto Bertoni et Marco Dorigo. Implicit parallelism in genetic algorithms. *Artificial Intelligence*, 61(2):307–314, 1993.

[BD95] James Bowen et Gerry Dozier. Solving constraint satisfaction problems using a genetic/systematic search hybrid that realizes when to quit. In *[Esh95]*, pages 122–129, 1995.

[BDF97] J. Christopher Beck, Andrew J. Davenport, et Mark S. Fox. Five pitfalls of empirical scheduling research, 1997. Principles and Practice of Constraint Programming.

[BE95] Thang Nguyen Bui et Paul H. Eppley. A hybrid genetic algorithm for the maximum clique problem. In *[Esh95]*, pages 478–484, 1995.

[Bea90] J.E. Beasley. OR-Library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41:1069–1072, 1990. available via anonymous ftp on `mscmga.ms.ic.ac.uk:/pub`.

[BEW95] Edmund K. Burke, Dave G. Elliman, et Rupert F. Weare. A hybrid genetic algorithm for highly constrained timetabling problems. In *[Esh95]*, pages 605–610, 1995.

[BF94] Lawrence Bull et Terence C. Fogarty. An evolution strategy and genetic algorithm hybrid : an initial implementation and first results. In *[Fog94a]*, pages 95–102, 1994.

[BFVY96] F. J. Budinsky, M. A. Finnie, J. M. Vlissides, et P. S. Yu. Automatic code generation from design patterns. *Object technology*, 35(2):14, 1996.

[BHPT98] Vincent Bachelet, Zouhir Hafidi, Philippe Preux, et El-Ghazali Talbi. Vers la coopération des méta-heuristiques. *Calculateurs parallèles, réseaux et systèmes répartis*, 10(2):211–223, Avril 1998.

[BHS97] Thomas Bäck, Ulrich Hammel, et Hans-Paul Schwefel. Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17, Avril 1997.

[BJ96] Alex Van Breedam et Gerrit K. Janssens. Rule-based job-shop scheduling via decision tables. In *[FUC96]*, pages 286–290, Septembre 1996.

[BLGR99] K. S. Barber, T. H. Liu, A. Goel, et S. Ramaswamy. Flexible reasoning using sensible agent-based systems : a case study in job flow scheduling. *Production Planning & Control*, 10(7):606–615, 1999. ISSN: 0953-7287.

[BM95] Thang Nguyen Bui et Byung Ro Moon. On multi-dimensional encoding/crossover. In *[Esh95]*, pages 49–56, 1995.

[BM96] Marc Bourcerie et Stéphane Marteau. An hybrid flow-shop modelling by Z/pZ-coloured petri net. In *[FUC96]*, pages 311–314, Septembre 1996.

[BMK96] Christian Bierwirth, Dirk C. Mattfeld, et Herbert Kopfer. On permutation representations for scheduling problems. In *[VERS96]*, pages 310–318, 1996.

[BMMM98] William J. Brown, Raphael C. Malveau, Hays W. McCormick, et Thomas J. Mowbray. *Anti Patterns - Refactoring Software, Architectures and Projects in Crisis.* Wiley Computer Publishing, 1998. ISBN: 0-471-19713-0.

[Boo87] Lashon Booker. Improving search in genetic algorithms. In *[Dav87]*, pages 61–73. Morgan Kaufmann, San Mateo, CA, USA, 1987.

[BP96] Philippe Baptiste et Claude Le Pape. A constraint-based branch-and-bound algorithm for preemptive job-shop scheduling. In *[FUC96]*, pages 24–27, Septembre 1996.

[BPT96] Vincent Bachelet, Philippe Preux, et El-Ghazali Talbi. Parallel hybrid meta-heuristics: Application to the quadratic assignment problem. In *Proceedings of the Parallel Optimization Colloquium*, pages 233–242, Versailles, France, Avril 1996.

[BPT97] Vincent Bachelet, Philippe Preux, et El-Ghazali Talbi. The landscape of the quadratic assignment problem and local search methods. In *Proceedings of the Tenth European Chapter on Combinatorial Optimization*, page 3, Mai 1997. (one page summary).

[Bra96] Marcello Braglia. Manufacturing cell design with size constraints: A multi perturbation simulated annealing approach. In *[FUC96]*, pages 328–331, Septembre 1996.

[BRJ99] Grady Booch, James Rumbaugh, et Ivar Jacobson. *The Unified Modeling Language - User Guide.* Addison Wesley, 1999. ISBN: 0-201-57168-4.

[Bru93] Ralf Bruns. Direct chromosome representation and advanced genetic operators for production scheduling. In *[For93]*, pages 352–359, 1993.

[Bru97] Michael Bruyere. Espaces de recherche des problèmes NP-durs. Master's thesis, Université des Sciences et Technologies de Lille 1, Villeneuve d'Ascq, 1997. DEA thesis, Laboratoire d'Informatique Fondamentale de Lille.

[BT94] Roberto Battiti et Giampietro Tecchiolli. The reactive tabu search. *ORSA Journal on Computing*, 6(2):126–140, 1994.

[BUMK91] Sugato Bagchi, Serdar Uckun, Yutaka Miyabe, et Kazuhiko Kawamura. Exploring problem-specific recombination operators for job shop scheduling. In *[BB91]*, pages 10–17, 1991.

[BWJ90] Steven Beaty, Darrell Whitley, et Gearold Johnson. Motivation and framework for using genetic algorithms for microcode compaction. In *Proceedings of the 23rd Annual Workshop and Symposium. MICRO 23. Microprogramming and Microarchitecture*, pages 117–124, Orlando, FL, USA, 27-29 Novembre 1990. IEEE Computer Society Press, Los Alamitos, CA. IEEE Cat. No. 90TH0341-8.

[BWM97] David Brittain, Jon Sims Williams, et Christopher Mc Mahon. A genetic algorithm approach to planning the telecommunications access network. In *[Bäc97]*, pages 623–628, 1997.

[Cal99] Patrice Roger Calégari. *Parallelization of population-based evolutionary algorithms for combinatorial optimization problems.* PhD thesis, École Polytechnique Fédérale de Lausanne, Lausanne, Septembre 1999.

[CC88] J. Carlier et P. Chrétienne. *Problèmes d'ordonnancement, modélisation, complexité, algorithmes.* Masson, 1988. ISBN: 2-225-81275-6.

[CCP94] Haoxun Chen, Chengbin Chu, et Jean-Marie Proth. Job shop scheduling using lagrangian relaxation, 1994. Rapport de recherche de l'INRIA, à paraître, et proposé pour publication dans IEEE Transactions on Robotics and Automation.

[CD87] Susan Coombs et Lawrence Davis. Genetic algorithms and communication link speed design : Constraints and operators. In *[Gre87a]*, pages 257–260, 1987.

[CDM90] Alberto Colorni, Marco Dorigo, et Vittorio Maniezzo. Genetic algorithms and highly constrained problems: The time-table case. In *[SM91]*, pages 55–59, 1990.

[CDMT94] Alberto Colorni, Marco Dorigo, Vittorio Maniezzo, et Marco Trubian. Ant system for jobshop scheduling. *Belgian Journal of Operations Research, Statistics and Computer Science*, 1994.

[CF94] Berthe Y. Choueiry et Boi Faltings. A decomposition heuristic for resource allocation. In *[Eib94]*, pages 585–589, 1994.

[Cha99] Philippe Chassaing. How many probes are needed to compute the maximum of a random walk ? *Stochastic processes and their applications*, 81:129–153, Sep/Nov 1999.

[Chu92a] Chengbin Chu. A branch-and-bound algorithm to minimize total flow time with unequal release dates. *Naval Research Logistics*, 39:859–875, 1992.

[Chu92b] Chengbin Chu. A branch-and-bound algorithm to minimize total tardiness with different release dates. *Naval Research Logistics*, 39:265–283, 1992.

[Chu95a] Chengbin Chu. A branch-and-bound approach for earliness-tardiness scheduling problems with different due dates, 1995. Rapport de recherche de l'INRIA, à paraître, et proposé pour publication dans Operations Research.

[Chu95b] Chengbin Chu. A new class of scheduling criteria and their optimization, 1995. à paraître.

[Chu95c] Chengbin Chu. *Ordonnancement de la Production : Approches Théoriques Nouvelles.* Thèse d'habilitation à diriger des recherches, Université de Metz, Metz, France, Juin 1995.

[CL95a] Yves Caseau et François Laburthe. Disjunctive scheduling with task intervals. Rapport technique LIENS-95-25, LIENS, 45 rue d'Ulm, 75230 Paris (France), Juillet 1995.

[CL95b] Yves Caseau et François Laburthe. Jobshop scheduling with task interval. In *International Workshop on Combinatorics and Computer Science (CCS'95)*, pages 19–21, Brest, France, Juillet 1995.

[CLR94] Thomas Cormen, Charles Leiserson, et Ronald Rivest. *Introduction à l'algorithmique.* Dunod, 1994. ISBN: 2-1000-3128-7.

[CM91] Hugh M. Cartwright et Gregory F. Mott. Looking around: Using clues from the data space to guide genetic algorithm searches. In *[BB91]*, pages 108–114, 1991.

[CM95a] Ricardo Corréa et Grégory Mounié. A parallel general branch-and-bound library and applications. In *International Workshop on Combinatorics and Computer Science (CCS'95)*, pages 30–32, Brest, France, Juillet 1995.

[CM95b] Rob Craighurst et Worthy Martin. Enhancing ga performance through crossover prohibitions based on ancestry. In *[Esh95]*, pages 130–135, 1995.

[CM98] Christine Crisan et Heinz Mühlenbein. The breeder genetic algorithm for frequency assignment. In *[EBSS98]*, pages 897–906, Septembre 1998.

[CMMR87] A. Corona, M. Marchesi, C. Martini, et S. Ridella. Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm. *ACM Transactions on Mathematical Software*, 13(3):262–280, Septembre 1987.

[CMMT97] Van-Dat Cung, Thierry Mautor, Philippe Michelon, et Andréa Tavares. Improving the efficiency of scatter search. In *[INR97]*, page 19, 1997. (one page summary).

[CMMT98] Van-Dat Cung, Thierry Mautor, Philippe Michelon, et Andréa Tavares. Recherche dispersée pour le problème de l'affectation quadratique. In *Premier congrès de la Société Française de Recherche Opérationnelle et Aide à la Décision (ROADF'98)*, page 151, Paris, France, Janvier 1998. (one page abstract).

[CN96] Jacques Carlier et Emmanuel Neron. A new branch and bound method for solving the resource constrained project scheduling problem. In *[FUC96]*, pages 255–258, Septembre 1996.

[COC97] A. Gaspar Cunha, Pedro Oliveira, et José A. Covas. Use of genetic algorithms in multicriteria optimization to solve industrial problems. In *[Bäc97]*, pages 682–688, 1997.

[CP89] J. Carlier et E. Pinson. An algorithm for solving the job-shop problem. *Management Science*, 35(2):164–176, 1989.

[CP96] Chengbin Chu et Jean-Marie Proth. *L'ordonnancement et ses applications*. Masson, 1996. ISBN: 2-225-85193-X.

[CPP92] Chengbin Chu, Marie-Claude Portmann, et Jean-Marie Proth. A splitting-up approach to simplify job-shop scheduling problem. *International Journal of Production Research*, 30(4):859–870, 1992.

[CPP95] Christophe Caux, Henri Pierreval, et Marie-Claude Portmann. Les algorithmes génétiques et leur application aux problèmes d'ordonnancement. *Automatique, Productique, Informatique Industrielle*, 29(4–5):409–443, 1995.

[CR99] Karunakaran Chakravarthy et Chandrasekharan Rajendran. A heuristic for scheduling in a flowshop with the bicriteria of makespan and maximum tardiness minimization. *Production Planning & Control*, 10(7):707–714, 1999. ISSN: 0953-7287.

[CRF94] Dave Corne, Peter Ross, et Hsiao-Lan Fang. Fast practical evolutionary timetabling. In *[Fog94a]*, pages 250–263, 1994.

[CS89] Gary A. Cleveland et Stephen F. Smith. Using genetic algorithms to schedule flow shop releases. In *[Sch89]*, pages 160–169, 1989.

[CT94] Hugh M. Cartwright et Andrew L. Tuson. Genetic algorithms and flowshop scheduling: towards the development of a real-time process control system. In *[Fog94a]*, pages 277–290, 1994.

[DAS97] Berna Dengiz, Fulya Altiparmak, et Alice E. Smith. Local search genetic algorithm for optimization of highly reliable communications networks. In *[Bäc97]*, pages 650–657, 1997.

[DASF94] Daniel Delahaye, Jean-Marc Alliot, Marc Schoenauer, et Jean-Loup Farges. Genetic algorithms for air traffic assignment. In *[Eib94]*, pages 33–37, 1994.

[Dav85] Lawrence Davis. Job shop scheduling with genetic algorithms. In *[Gre85]*, pages 136–140, 1985.

[Dav87] Lawrence Davis, éditeur. *Genetic Algorithms and Simulated Annealing.* Research Notes in Artificial Intelligence. Morgan Kaufmann, San Mateo, CA, USA, 1987. ISBN: 0-934613-44-3.

[Dav91a] Yuval Davidor. A naturally occuring niche & species phenomenon : The model and first results. In *[BB91]*, pages 257–263, 1991.

[Dav91b] Lawrence Davis, éditeur. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, USA, 1991. ISBN: 0-442-00173-8.

[DBB94] Gerry Dozier, James Bowen, et Dennis Bahler. Constraint processing using heuristic microgenetic algorithms. In *[Eib94]*, 1994.

[dD93] Benoît Dupont de Dinechin. Simplex scheduling : More than lifetime-sensitive instruction scheduling, 1993.

[De 95] Kenneth A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan, Ann Arbor, 1995. Dissertation Abstracts International 36(10), 5140B; UMI 76-9381.

[DG94] Jürgen Dorn et Mario Girsch. Genetic operators based on constraint repair. In *[Eib94]*, 1994.

[DG95] Marco Dorigo et Luca Maria Gambardella. ANT-Q a cooperative learning approach to combinatorial optimization. Rapport technique Technical Report 95-01, IRIDIA Université Libre de Bruxelles, Bruxelles, Belgium, 1995.

[DG96] Marco Dorigo et Luca Maria Gambardella. A study of some properties of ant-q. In *[VERS96]*, pages 656–665, 1996.

[DG97] Marco Dorigo et Luca Maria Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, Avril 1997.

[DH94] Raphaël Dorne et Jin-Kao Hao. Nouveaux opérateurs génétiques appliqués à SAT. In *[ALRS94]*, pages 143–151, 1994.

[DH98] Raphaël Dorne et Jin-Kao Hao. A new genetic local search algorithm for graph coloring. In *[EBSS98]*, pages 745–754, Septembre 1998.

[DM92] Marco Dorigo et Vittorio Maniezzo. Parallel genetic algorithms: Introduction and overview of current research. In *[Ste92]*, pages 5–42, 1992.

[DN92] Eric Davalo et Patrick Naïm. *Des Réseaux de Neurones.* Eyrolles, second edition, 1992. ISBN: 2-212-08137-5.

[DP92] U. Dorndorf et E. Pesh. Evolution-based learning in a job shop scheduling environment. Rapport technique RM 92-019, Rijksuniversiteit Limburg, The Netherlands, 1992.

[DP95] Lamia Djerid et Marie-Claude Portmann. Comment entrecroiser des procédures par séparation et évaluation et des algorithmes génétiques : application à des problèmes d'ordonnancement à contraintes disjonctives. In *Francoro 95*, pages 84–85, Juin 1995. (one page summary).

[DP96] Rutvik Desai et Rajendra Patil. Salo : combining simulated annealing and local optimization for efficient global optimization. In *Proceedings of the* 9^{th} *Florida Artificial Intelligence Research Symposium (FLAIRS)*, pages 233–237, Juin 1996.

[DPF+97] David Duvivier, Philippe Preux, Cyril Fonlupt, Denis Robilliard, et El-Ghazali Talbi. The fitness function and its impact on local search methods. Rapport technique LIL-1997-04, Laboratoire d'Informatique du Littoral, Calais, France, Septembre 1997.

[DPF+98] David Duvivier, Philippe Preux, Cyril Fonlupt, Denis Robilliard, et El-Ghazali Talbi. Impact de la fonction objectif sur les performances des algorithmes itératifs de recherche locale. In *Premier congrès de la Société Française de Recherche Opérationnelle et Aide à la Décision (ROADF'98)*, page 137, Paris, France, Janvier 1998. (one page abstract).

[DPT95a] David Duvivier, Philippe Preux, et El-Ghazali Talbi. Algorithmes génétiques parallèles pour l'optimisation : application aux problèmes de job-shop et d'affectation quadratique. In *Francoro 95*, pages 85–86, Juin 1995. (one page summary).

[DPT95b] David Duvivier, Philippe Preux, et El-Ghazali Talbi. Application des algorithmes génétiques au problème du job-shop-scheduling. Rapport technique LIL-1995-03, Laboratoire d'Informatique du Littoral, Calais, France, Avril 1995.

[DPT95c] David Duvivier, Philippe Preux, et El-Ghazali Talbi. Parallel genetic algorithms for optimization and application to NP-complete problem solving. In *International Workshop on Combinatorics and Computer Science (CCS'95)*, pages 41–43, Brest, France, Juillet 1995.

[DPT95d] David Duvivier, Philippe Preux, et El-Ghazali Talbi. Stochastic algorithms for optimization and application to the job-shop-scheduling problem. Rapport technique LIL-1995-05, Laboratoire d'Informatique du Littoral, Calais, France, Septembre 1995.

[DPT96a] David Duvivier, Philippe Preux, et El-Ghazali Talbi. Climbing up NP-hard hills. In *[VERS96]*, pages 574–583, Septembre 1996.

[DPT96b] David Duvivier, Philippe Preux, et El-Ghazali Talbi. Genetic algorithms applied to the job-shop scheduling problem. In *[FUC96]*, pages 28–31, Septembre 1996. An extended version is available as report LIL-1995-04 from the Laboratoire d'Informatique du Littoral.

[DPT+98] David Duvivier, Philippe Preux, El-Ghazali Talbi, Cyril Fonlupt, et Denis Robilliard. The fitness function and its impact on local search methods. In *Proceedings of the conference IEEE Systems, Man, and Cybernetics*, pages 2478–2483, San Diego, USA, Octobre 1998.

[DS87] Lawrence Davis et Martha Steenstrup. Genetic algorithms and simulated annealing : An overview. In *[Dav87]*, pages 1–11. Morgan Kaufmann, San Mateo, CA, USA, 1987.

[DSM94] Y. Davidor, H-P. Schwefel, et R. Maenner, éditeurs. *Parallel Problem Solving from Nature – PPSN III*, Jerusalem, Israel, Octobre 1994. Springer-Verlag, Berlin. Lecture Notes in Computer Science, vol. 866.

[Duv94] David Duvivier. Algorithmes génétiques sur calculateurs parallèles. Master's thesis, Université des Sciences et Technologies de Lille 1, Villeneuve d'Ascq, 1994. DEA thesis, Laboratoire d'Informatique Fondamentale de Lille.

[DW94] John Dzubera et Darrell Whitley. Advanced correlation analysis of operators for the traveling salesman problem. In *[DSM94]*, pages 68–77, 1994.

[DYN93] Yuval Davidor, Takeshi Yamada, et Ryohei Nakano. The ECOlogical framework II: Improving GA performance at virtually zero cost. In *[For93]*, pages 171–176, 1993.

[EAVA97] H.M.M. Ten Eikelder, B.J.M. Aarts, M.G.A. Verhoeven, et E.H.L. Aarts. Sequential and parallel local search algorithms for job shop scheduling. In *[INR97]*, pages 75–80, 1997.

[EBSS98] Agoston E. Eiben, Thomas Bäck, Marc Schoenauer, et Hans-Paul Schwefel, éditeurs. *Parallel Problem Solving from Nature – PPSN V*, Berlin, Germany, Septembre 1998. Springer-Verlag, Berlin. Lecture Notes in Computer Science, vol. 1498, ISSN: 0302-9743.

[Edm98] Bruce Edmonds. Meta-genetic programming: Co-evolving the operators of variation. Rapport technique CPM Report No. 98-32, Centre for Policy Modelling, Manchester Metropolitan University, Manchester, UK, Janvier 1998.

[Eib94] A. Eiben, éditeur. *Proceedings of the ECAI'94 workshop on Applied Genetic and Other Evolutionary Algorithms*, Amsterdam, The Netherlands, Août 1994. ECCAI, John Wiley and Sons.

[EL99] Patrick Esquirol et Pierre Lopez. *L'ordonnancement*. Economica, 1999. ISBN: 2-7178-3798-1.

[EM93] Fred F. Easton et Nashat Mansour. A distributed genetic algorithm for employee staffing and scheduling problems. In *[For93]*, pages 360–367, 1993.

[ER96] Ian Robert East et Jon Rowe. Effects of isolation in a distributed population genetic algorithm. In *[VERS96]*, pages 408–419, 1996.

[ERR94] A.E. Eiben, P-E. Raué, et Zs. Ruttkay. On the notion of constrained problems: a ga point of view. In *[Eib94]*, 1994.

[ES91] Larry J. Eshelman et J. David Schaffer. Preventing premature convergence in genetic algorithms by preventing incest. In *[BB91]*, pages 115–122, 1991.

[Esh95] Larry J. Eshelman, éditeur. *Proceedings of the Sixth International Conference on Genetic Algorithms*, Pittsburgh, PA, USA, Juillet 1995. Morgan Kaufmann, San Mateo, CA, USA. ISBN: 1-55860-370-0.

[EvdH97] A.E. Eiben et J.K. van der Hauw. Adaptive penalties for evolutionary graph coloring. In *[HLR+97]*, pages 251–262, 1997.

[EvHMS98] A.E. Eiben, J.I. van Hemert, E. Marchiori, et A.G. Steenbeek. Solving binary constraint satisfaction problems using evolutionary algorithms with an adaptive fitness function. In *[EBSS98]*, pages 201–210, Septembre 1998.

[Evo97] EvoStim. The state of the art in evolutionary approaches to timetabling and scheduling. Rapport technique, EvoStim : The EVONET Working Group on Evolutionary Scheduling and Timetabling, 1997.

[Fan92] Hsiao-Lan Fang. Investigating genetic algorithms for scheduling. Master's thesis, University of Edinburgh, 1992.

[Fan94] Hsiao-Lan Fang. *Genetic Algorithms in Timetabling and Scheduling*. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, Septembre 1994.

[FB91] E. Falkenauer et S. Bouffouix. A genetic algorithm for job shop. In *Proceedings 1991 IEEE International Conference on Robotics and Automation*, pages 824–829, Sacramento, CA, Avril 1991. IEEE Computer Society Press, Los Alamitos, CA. Cat. No. 91CH2969-4.

[FB98] Enrico Faggioli et Carlo Alberto Bentivoglio. Heuristic and exact methods for the cutting sequencing problem. *European Journal of Operational Research*, 110:564–575, 1998.

[FD98] Clarisse Flipo-Dhaenens. *Optimisation d'un réseau de production et de distribution*. PhD thesis, Institut National Polytechnique de Grenoble, Octobre 1998.

[FF93] Carlos M. Fonseca et Peter J. Fleming. Genetic algorithms for multiobjective optimization : Formulation, discussion and generalization. In *[For93]*, pages 416–423, 1993.

[FF95] Charles C. Fleurent et Jacques A. Ferland. Algorithmes génétiques hybrides pour l'optimisation combinatoire. *RAIRO*, 1995.

[FGL96] Gérard Fleury, Michel Gourgand, et Philippe Lacomme. Stochastic algorithms to solve problems in transport systems. In *[FUC96]*, pages 360–363, Septembre 1996.

[FL99] Olivier François et Christian Lavergne. Plans d'expériences pour l'évaluation d'algorithmes évolutionnaires et la constitution de classes de référence. Rapport technique 3601, INRIA Rhone-Alpes, Janvier 1999. ISSN: 0249-6399.

[FL00a] Rémy Fannader et Hervé Leroux. *UML Principes de mise en oeuvre.* Dunod, 2000. ISBN: 2-100-05309-4.

[FL00b] Rémy Fannader et Hervé Leroux. *UML Principes de modélisation.* Dunod, 2000. ISBN: 2-100-04650-0.

[Fle95] Gérard Fleury. Applications de méthodes stochastiques inspirées du recuit simulé à des problèmes d'ordonnancement. *Automatique, Productique, Informatique Industrielle*, 29(4–5):445–470, 1995.

[FM91] B.R. Fox et M.B. McMahon. Genetic operators for scheduling problems. In *[Raw91]*, pages 284–300, 1991.

[Fog94a] Terence C. Fogarty, éditeur. *Proceedings of the AISB Workshop on Evolutionary Computing*, Lecture Notes in Computer Science, vol 865, Leeds, UK, Avril 1994. Springer-Verlag. ISBN: 3-540-58483-8.

[Fog94b] David B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1):3–14, Janvier 1994.

[For93] Stephanie Forrest, éditeur. *Proceedings of the Fifth International Conference on Genetic Algorithms*, Urbana-Champaign, IL, USA, Juillet 1993. Morgan Kaufmann, San Mateo, CA, USA. ISBN: 1-55860-299-2.

[FPRT98] Cyril Fonlupt, Philippe Preux, Denis Robilliard, et El-Ghazali Talbi. Paysages des problèmes NP-durs et métaheuristiques. In *Premier congrès de la Société Française de Recherche Opérationnelle et Aide à la Décision (ROADF'98)*, page 22, Paris, France, Janvier 1998. (one page abstract).

[FPT97] Cyril Fonlupt, Philippe Preux, et El-Ghazali Talbi. Paysages adaptatifs et algorithmes évolutifs. In *Evolutionary Algorithms: From Theory to Applications*, Marseille, France, Juin 1997.

[FRC93] Hsiao-Lan Fang, Peter Ross, et Dave Corne. A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems. In *[For93]*, pages 488–493, 1993.

[FRC94] Hsiao-Lan Fang, Peter Ross, et Dave Corne. A promising hybrid ga / heuristic approach for open-shop scheduling problems. In *[Eib94]*, pages 590–594, 1994.

[FRP97a] Cyril Fonlupt, Denis Robilliard, et Philippe Preux. Fitness landscape and the behavior of heuristics. In *[HLR+97]*, pages 321–329, 1997.

[FRP97b] Cyril Fonlupt, Denis Robilliard, et Philippe Preux. Preventing premature convergence via cooperating genetic algorithms. In *Proceedings of Mendel'97*, pages 50–55, Juin 1997.

[FRP98] Cyril Fonlupt, Denis Robilliard, et Philippe Preux. A bit-wise epistasis measure for binary search spaces. In *[EBSS98]*, pages 47–56, Septembre 1998.

[FRPT99] Cyril Fonlupt, Denis Robilliard, Philippe Preux, et El-Ghazali Talbi. Fitness landscapes and performance of meta-heuristics. In *Meta-heuristics – Advances and Trends in Local Search Paradigms for Optimization*, pages 255–266. Kluwer Academic Press, 1999.

[FS98] Uriel Feige et Christian Scheideler. Improved bounds for acyclic job shop scheduling. Rapport technique Technical Report 98-10, Février 1998.

[FTP97] Cyril Fonlupt, El-Ghazali Talbi, et Philippe Preux. Fitness landscape and performance of meta-heuristics. In *[INR97]*, pages 45–46, 1997.

[FUC96] FUCAM, éditeur. *Proceedings of the Workshop on Production Planning and Control*, Mons, Belgium, Septembre 1996.

[FVC95] T. C. Fogarty, F. Vavak, et P. Cheng. Use of the genetic algorithm for load balancing of sugar beet presses. In *[Esh95]*, pages 617–624, 1995.

[GBH+91] Paula S. Gabbert, Donald E. Brown, Christopher L. Huntley, Bernard P. Markowicz, et David E. Sappington. A system for learning routes and schedules with genetic algorithms. In *[BB91]*, pages 430–436, 1991.

[GBP96] Alfred B. Gembeh, S. K. Banerjee, et K. V. Pandya. Short-term capacity control for a dynamic job shop. In *[FUC96]*, pages 32–35, Septembre 1996.

[Ghe92] Fatima Ghedjati. Heuristiques pour le problème JOB-SHOP avec machines non reliées en parallèle. In *Colloque International – Méthodes et Outils d'Aide à la Décision (MOAD'92)*, pages 83–88, 1992.

[Ghe95] Fatima Ghedjati. Genetic algorithm for the generalized job-shop scheduling problem, 1995.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, et John Vlissides. *Design Patterns CD - Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995. ISBN: 0-201-30953-X (CDROM).

[Gil97] Jaysen Gillespie. A genetic algorithm solution to the project selection problem using static and dynamic fitness functions. In *[Koz97]*, pages 76–85, 1997.

[GJ79] Michael R. Garey et David S. Johnson. *Computers and Intractability; A Guide to the theory of NP-Completeness*. W.H. Freeman and Company, 1979. ISBN: 0-7167-1045-5.

[Glo86] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Comput. Ops. Res.*, 13(5):533–549, 1986.

[Glo89a] Fred Glover. Tabu search – part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[Glo89b] Fred Glover. Tabu search – part II. *ORSA Journal on Computing*, 2(1):4–31, 1989.

[Glo95] Fred Glover. Tabu search fundamentals and uses, 1995. Survey revised and expanded: April 1995.

[Glo97] Fred Glover. A template for scatter search and path relinking. In *[HLR+97]*, pages 10–39, 1997.

[GM95] Michel Gondran et Michel Minoux. *Graphes et algorithmes*. Eyrolles, 1995. ISSN: 0399-4198.

[GOD+98] C. Gil, J. Ortega, A.F. Diaz, M.G. Montoya, et A. Prieto. Load balancing in parallel circuit testing with annealing-based and genetic algorithms. In *[EBSS98]*, pages 835–844, Septembre 1998.

[Gol87] David Goldberg. Simple genetic algorithms and the minimal deceptive problem. In *[Dav87]*, pages 74–88. Morgan Kaufmann, San Mateo, CA, USA, 1987.

[Gol89] David Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989. ISBN: 0-201-15767-5.

[Gol94] David E. Goldberg. *Algorithmes Génétiques, exploration, optimisation et apprentissage automatique*. Addison-Wesley, 1994. ISBN: 2-87908-054-1.

[GOT93] GOThA. Groupe d'ordonnancement théorique et appliqué. les problèmes d'ordonnancement. *Operations Research*, 27(1):77–150, 1993.

[GP97] Fatima Ghedjati et Jean-Charles Pomerol. Résolution du problème d'ordonnancement de type jobshop généralisé par des heuristiques dynamiques. Rapport technique lip6.1997.005, LAFORIA-IBP, 1997.

[GPP95] F. Ghedjati, J. C. Pomerol, et M.-C. Portmann. Algorithmes génétiques pour le problème d'ordonnancement de type jobshop généralisé : méthodes par brassage d'heuristiques. In *Francoro 95*, pages 86–87, Juin 1995. (one page summary).

[GPSS96] Leslie Ann Goldberg, Mike Paterson, Aravind Srinivasan, et Elizabeth Sweedyk. Better approximation guarantees for job-shop scheduling, 1996. available via anonymous ftp on `ftp.dcs.warwick.ac.uk:/pub/reports/rr/312/all.ps.Z`.

[Gre85] J. J. Grefenstette, éditeur. *Proceedings of the First International Conference on Genetic Algorithms*, Pittsburgh, PA, USA, Juillet 1985. Lawrence Erlbaum Associates: Hillsdale, New-Jersey. ISBN: 0-805-80426-9.

[Gre86] John Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 16(1):122–128, Janvier 1986.

[Gre87a] J.J. Grefenstette, éditeur. *Proceedings of the Second International Conference on Genetic Algorithms*, MIT, Cambridge, MA, USA, Juillet 1987. Lawrence Erlbaum Associates: Hillsdale, New-Jersey. ISBN: 0-805-80159-6.

[Gre87b] John J. Grefenstette. Incorporating problem specific knowledge into genetic algorithms. In *[Dav87]*, pages 42–60. Morgan Kaufmann, San Mateo, CA, USA, 1987.

[Gre93] John J. Grefenstette. Deception considered harmful. In *[Whi93c]*, pages 75–91, 1993.

[Gru94] Frédéric Gruau. *Synthèse de réseaux de Neurones par Codage Cellulaire et Algorithmes Génétiques*. PhD thesis, LIP-IMAG, Lyon, France, 1994.

[GS78] Teofilo Gonzalez et Sartaj Sahni. Flowshop and jobshop schedules: Complexity and approximation. *Operations Research*, 26(1):36–52, 1978.

[GS97] Christine Gaspin et Thomas Schiex. Genetic algorithms for genetic mapping. In *[HLR+97]*, pages 142–152, 1997.

[GT69] B. Giffler et G. L. Thompson. Algorithms for solving production scheduling problems. *Operations Research*, 8:487–503, 1969.

[GV97] Jens Gottlieb et Nico Voss. Representations, fitness functions and genetic operators for the satisfiability problem. In *[HLR+97]*, pages 43–56, 1997.

[GW93] V. Scott Gordon et Darrell Whitley. Serial and parallel genetic algorithms as function optimizers. Rapport technique CS-93-114, Colorado State University; Department of Computer Science, Fort Collins, CO 80523, USA, Août 1993.

[Han94] Peter J. B. Hancock. An empirical comparison of selection methods in evolutionary algorithms. In *Proceedings of the AISB workshop on Evolutionary Computation*, Juillet 1994.

[HB91] William E. Hart et Richard K. Belew. Optimizing an arbitrary function is hard for the genetic algorithm. In *[BB91]*, pages 190–195, 1991.

[HD90] Wenxue Han et Pierre Dejax. Une heuristique pour le problème d'ordonnancement de type $n/m//F/C_{\max}$ avec la présence de machines goulots. *Operations Research*, 24(4):315–330, 1990.

[Her90] Michael Herdy. Application of the evolutionsstrategie to discrete optimization problems. In *[SM91]*, pages 188–192, 1990.

[Her91] Alain Hertz. Tabu search for large scale timetabling problems. *European Journal of Operational Research*, 54:39–47, 1991.

[HG94] Jeffrey Horn et David E. Goldberg. Genetic algorithm difficulty and the modality of fitness landscapes. In *[WV94]*, 1994.

[HH97] Karl-Werner Hansmann et Michael Hoeck. Production control of a flexible manufacturing system in a job shop environment. *International Transactions in Operational Research*, 4(5/6):341–351, Sep/Nov 1997. ISSN: 0969-6016.

[HK94] Anthony M. Hill et Sung-Mo (Steve) Kang. Genetic algorithm based design optimization of CMOS VLSI circuits. In *[DSM94]*, pages 546–555, 1994.

[HKB94] William E. Hart, Thomas E. Kammeyer, et Richard K. Belew. The role of development in genetic algorithms. In *[WV94]*, 1994.

[HLM98] Francisco Herrera, Manuel Lozano, et Claudio Moraga. Hybrid distributed real-coded genetic algorithms. In *[EBSS98]*, pages 603–612, Septembre 1998.

[HLP+87] M. R. Hilliard, G. E. Liepins, Mark Palmer, Michael Morrow, et Jon Richardson. A classifier-based system for discovering scheduling heuristics. In *[Gre87a]*, pages 231–235, 1987.

[HLR+97] J-K. Hao, E. Lutton, E. Ronald, M. Schoenauer, et D. Snyers, éditeurs. *Proceedings of Evolution Artificielle 97 EA '97*, volume 1363, Nimes, France, Octobre 1997. Springer-Verlag, Lecture Notes in Computer Science.

[HM91] Philip Husbands et Frank Mill. Simulated co-evolution as the mechanism for emergent planning and scheduling. In *[BB91]*, pages 264–270, 1991.

[HM95] Wim Hordijk et Bernard Manderick. The usefulness of recombination, Mars 1995.

[HM97] P. Hansen et N. Mladenović. An introduction to variable neighbourhood search. In *[INR97]*, pages 9–10, 1997.

[HMW90] Philip Husbands, Frank Mill, et Stephen Warrington. Genetic algorithms, production plan optimisation and scheduling. In *[SM91]*, pages 80–84, 1990.

[HN93] Jeffrey Horn et Nicholas Nafpliotis. Multiobjective optimization using the niched pareto genetic algorithm. Rapport technique IlliGALs report 93005, IlliGALs, department of General Engineering, University of Illinois at Urbana-Champaign, 117 Transportation Building, 104 South Mathews Avenue, Urbana, IL 61801, USA, Juillet 1993.

[Hol75] John H. Holland. *Adaptation in Natural and Artificial Systems.* Michigan Press University, Ann Arbor, MI, 1975.

[Hol92a] John Holland. Les algorithmes génétiques. *Pour la Science*, 179:44–51, Septembre 1992.

[Hol92b] John H. Holland. *Adaptation in Natural and Artificial Systems.* A Bradford Book. MIT Press, Cambridge, MA, USA, second edition, 1992. ISBN: 0-262-58111-6.

[Hop88] John J. Hopfield. Artificial neural networks. *IEEE Circuits and Devices*, 4(5):3–10, Septembre 1988.

[Hor94] Wim Hordijk. Population flow on fitness landscapes. Master's thesis, Erasmus University Rotterdam, Rotterdam, The Netherlands, Août 1994.

[Hor96] Wim Hordijk. A measure of landscapes. *Evolutionary Computation*, 4(4), 1996. ISSN: 1063-6560.

[HR98] Emma Hart et Peter Ross. A heuristic combination method for solving job-shop scheduling problems. In *[EBSS98]*, pages 844–854, Septembre 1998.

[HTdW97] Alain Hertz, Eric Taillard, et Dominique de Werra. A tutorial on tabu search. In E. Aarts et J. K. Lenstra, éditeurs, *Local Search in Combinatorial Optimization*, pages 121–136. J. Wiley & Sons Ltd., 1997.

[HTG96] Zouhir Hafidi, El-Ghazali Talbi, et Jean-Marc Geib. MARS: Un ordonnanceur adaptatif d'applications parallèles dans un environnement multi-utilisateurs. In *[ren96]*, pages 37–40, 1996.

[HTG97] Zouhir Hafidi, El-Ghazali Talbi, et Jean-Marc Geib. Méta-systèmes : vers l'intégration des machines parallèles et les réseaux de stations hétérogènes. *Calculateurs parallèles*, 9(4):435–450, 1997.

[Hul90] Martin Hulin. Circuit partitioning with genetic algorithms using a coding scheme to preserve the structure of a circuit. In *[SM91]*, pages 75–79, 1990.

[Hul97] Martin Hulin. An optimal stop criterion for genetic algorithms : A bayesian approach. In *[Bäc97]*, pages 135–142, 1997.

[Hus93] Philip Husbands. Genetic algorithms for scheduling. In *AISB Quarterly, No. 89*, 1993.

[Hus94] Phil Husbands. Distributed coevolutionary genetic algorithms for multi-criteria and multi-constraint optimisation. In *[Fog94a]*, pages 150–165, 1994.

[HW95] Alain Hertz et Marino Widmer. La méthode tabou appliquée aux problèmes d'ordonnancement. *Automatique, Productique, Informatique Industrielle*, 29(4–5):353–378, 1995.

[HW96] Alain Hertz et Marino Widmer. An improved tabu search approach for solving the job shop scheduling problem with tooling constraints. *Discrete Applied Mathematics*, 65:319–345, 1996.

[IHI96] M. Mc Ilhagga, P. Husbands, et R. Ives. A comparison of optimization techniques for integrated manufacturing planning and scheduling. In *[VERS96]*, pages 604–613, 1996.

[IKB+97] Ilkka Ikonen, Anup Kumar, William E. Biles, Rammohan K. Ragade, et John C. Wissel. A genetic algorithm for packing three-dimensional non-convex objects having cavities and holes. In *[Bäc97]*, pages 591–598, 1997.

[IMT97] Hisao Ishibuchi, Tadahiko Murata, et Shigemitsu Tomioka. Effectiveness of genetic local search algorithms. In *[Bäc97]*, pages 505–512, 1997.

[INR97] INRIA, éditeur. *Proceedings of the 2nd International Conference on MetaHeuristics (MIC'97)*, Sophia-Antipolis, France, Juillet 1997. INRIA. ISBN: 2-726-11084-3.

[JB91] Donald R. Jones et Mark A. Beltramo. Solving partitioning problems with genetic algorithms. In *[BB91]*, pages 442–449, 1991.

[JF95a] Terry Jones et Stephanie Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *[Esh95]*, pages 184–192, 1995.

[JF95b] Terry Jones et Stephanie Forrest. Genetic algorithms and heuristic search. In *International Joint Conference on Artificial Intelligence*, Montreal, Canada, Août 1995.

[JGSB92] Wilfried Jakob, Martina Gorges-Schleuter, et Christian Blume. Application of genetic algorithms to task planning and learning. In *[MM92]*, pages 291–300, 1992.

[JH] Jeffrey A. Joines et Christopher R. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's.

[JM74] Lynwood A. Johnson et Douglas C. Montgomery. *Operations Research in Production Planning, Scheduling, and Inventory Control.* John Wiley and Sons, 1974. ISBN: 0-471-44618-1.

[JM91a] Cezary Z. Janikov et Zbigniew Michalewicz. An experimental comparison of binary and floating point representations in genetic algorithms. In *[BB91]*, pages 31–36, 1991.

[JM91b] Cezary Z. Janikov et Zbigniew Michalewicz. Handling constraints in genetic algorithms. In *[BB91]*, pages 151–157, 1991.

[JM99] A.S. Jain et S. Meeran. Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research*, 113:390–434, 1999.

[Jon94] Terry Jones. A description of Holland's royal road function, Juin 1994.

[Jon95] Terry Jones. Crossover, macromutation, and population-based search. In *[Esh95]*, pages 73–80, 1995.

[JPY88] David S. Johnson, Christos H. Papadimitriou, et Mihalis Yannakakis. How easy is local search ? *Journal of Computer and System Sciences*, 37:79–100, 1988.

[JS89] Kenneth A. De Jong et William M. Spears. Using genetic algorithms to solve NP-complete problems. In *[Sch89]*, pages 124–132, 1989.

[JSG89] Prasanna Jog, Jung Y. Suh, et Dirk Van Gucht. The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem. In *[Sch89]*, pages 110–115, 1989.

[Jul93] Kate Juliff. A multi-chromosome genetic algorithm for pallet loading. In *[For93]*, pages 467–473, 1993.

[JW93] Joe L. Blanton Jr. et Roger L. Wainwright. Multiple vehicle routing with time and capacity constraints using genetic algorithms. In *[For93]*, pages 452–459, 1993.

[JW94] Ari Juels et Martin Wattenberg. Stochastic hillclimbing as a baseline method for evaluating genetic algorithms. Rapport technique csd-94-834, University of California, 1994.

[Kal98] Leila Kallel. Inside ga dynamics : Ground basis for comparison. In *[EBSS98]*, pages 57–66, Septembre 1998.

[KBH94a] Sami Khuri, Thomas Bäck, et Jörg Heitkötter. An evolutionary approach to combinatorial optimization problems. In *Proceedings of the 1994 Computer Science Conference (CSC'94)*, Phoenix, AZ, Mars 1994. ACM Press.

[KBH94b] Sami Khuri, Thomas Bäck, et Jörg Heitkötter. The zero/one multiple knapsack problem and genetic algorithms. In *Proceedings of the 1994 ACM Symposium on Applied Computing*. ACM Press, Mars 1994.

[KCR95] Dae Gyu Kim, David Corne, et Peter Ross. Industrial plant pipe-route optimisation with genetic algorithms. In *[Esh95]*, pages 1012–1021, 1995.

[KDG92] Hillol Kargupta, Kalyanmoy Deb, et David E. Goldberg. Ordering genetic algorithms and deception. In *[MM92]*, pages 47–56, 1992.

[KF98] Wael Khatib et Peter J. Fleming. The stud ga : A mini revolution ? In *[EBSS98]*, pages 683–691, Septembre 1998.

[KH98] Dae Gyu Kim et Phil Husbands. Landscape changes and the performance of mapping based constraint handling methods. In *[EBSS98]*, pages 221–230, Septembre 1998.

[Khu94] Sami Khuri. Walsh and Haar functions in genetic algorithms. In *Proceedings of the 1994 ACM Symposium on Applied Computing*. ACM Press, Mars 1994.

[Kid93] Michelle D. Kidwell. Using genetic algorithms to schedule distributed tasks on a bus-based system. In *[For93]*, pages 368–374, 1993.

[KKN93] Takashi Kido, Hiroaki Kitano, et Masakazu Nakanishi. A hybrid search for genetic algorithms: Combining genetic algorithms, tabu search, and simulated annealing. In *[For93]*, page 647, 1993. (one page summary).

[KM98] Slawomir Koziel et Zbigniew Michalewicz. A decoder-based evolutionary algorithm for constrained parameter optimization problems. In *[EBSS98]*, pages 231–240, Septembre 1998.

[Kor71] Mana Kortas. *Programmation linéaire en nombres entiers : comparaison de méthodes*. PhD thesis, Université des Sciences et Technologies de Lille I, Lille, Mars 1971.

[KOY95] Shigenobu Kobayashi, Isao Ono, et Masayuki Yamamura. An efficient genetic algorithm for job shop scheduling problems. In *[Esh95]*, pages 506–511, 1995.

[Koz97] John R. Koza, éditeur. *Genetic Algorithms and Genetic Programming at Stanford 1997*, Standford, California, USA, 1997. Stanford Bookstore, Standford University. ISBN: 0-18205-981-2.

[KP98] S. Kazarlis et V. Petridis. Varying fitness functions in genetic algorithms : Studying the rate of increase of the dynamic penalty terms. In *[EBSS98]*, pages 211–220, Septembre 1998.

[KRC95] Kazuhiro Kado, Peter Ross, et Dave Corne. A study of genetic algorithm hybrids for facility layout problems. In *[Esh95]*, pages 498–505, 1995.

[KSV90] Berthold Kröger, Peter Schwenderling, et Oliver Vornberger. Parallel genetic packing of rectangles. In *[SM91]*, pages 160–164, 1990.

[KSV92] Berthold Kröger, Peter Schwenderling, et Oliver Vornberger. Parallel genetic packing on transputers. In *[Ste92]*, pages 151–185, 1992.

[Kur90] Frank Kursawe. A variant of evolution strategies for vector optimization. In *[SM91]*, pages 193–197, 1990.

[Lai00] Michel Lai. *UML La notation unifiée de modélisation objet - De java aux EJB*. Dunod, 2000. ISBN: 2-100-05023-0.

[Lan95] William B. Langdon. Pareto, population partitioning, price and genetic programming. Research Note RN/95/29, University College London, Dept. of Computer Science, University College London, Gower Street, London WC1E 6BT, UK, Avril 1995. Submitted to AAAI Fall 1995 Genetic Programming Symposium.

[LC95a] François Laburthe et Yves Caseau. Résolution des problèmes d'ordonnancement d'atelier en raisonnant sur des intervalles de tâches. In *Francoro 95*, page 16, Juin 1995. (one page summary).

[LC95b] Jens Lienig et James P. Cohoon. Genetic algorithms applied to the physical design of vlsi circuits : A survey. In *[Esh95]*, pages 839–848, 1995.

[Lev93] David M. Levine. A genetic algorithm for the set partitioning problem. In *[For93]*, pages 481–487, 1993.

[Lev94] David Levine. A parallel genetic algorithm for the set partitioning problem, Décembre 1994.

[LGWFP97] Shyh-Chang Lin, Erik D. Goodman, et III William F. Punch. A genetic approach to dynamic job shop scheduling problems. In *[Bäc97]*, pages 481–488, 1997.

[Lin92] Si-Eng Ling. Integrating genetic algorithms with a PROLOG assignment program as a hybrid solution for a polytechnic timetable problem. In *[MM92]*, pages 321–329, 1992.

[LK92] C. B. Lucasius et G. Kateman. Towards solving subset selection problems with the aid of the genetic algorithm. In *[MM92]*, pages 239–247, 1992.

[LPQ97] Gilbert Laporte, Jean-Yves Potvin, et Florence Quilleret. A tabu search heuristic using genetic diversification for the clustered traveling salesman problem. *Journal of Heuristics*, 2(3):187–200, 1997. ISSN: 1381-1231.

[LR97] Daniel H. Loughlin et S. Ranjithan. The neighborhood constraint method : A genetic algorithm-based multiobjective optimization technique. In *[Bäc97]*, pages 666–673, 1997.

[LRS98a] Marco Laumanns, Günter Rudolph, et Hans-Paul Schwefel. A spatial predator-prey approach to multi-objective optimization : A preliminary study. In *[EBSS98]*, pages 241–249, Septembre 1998.

[LRS+98b] Alexander Leonhardi, Wolfgang Reissenberger, Tim Schmelmer, Karsten Weicker, et Nicole Weicker. Development of problem-specific evolutionary algorithms. In *[EBSS98]*, pages 388–397, Septembre 1998.

[LS94] Xianzhang Lei et Lawrence H. Staib. A genetic algorithm for constrained optimization: Application to 3D medical image registration. In *[Eib94]*, 1994.

[LSS93] In Lee, Riyaz Sikora, et Michael J. Shaw. Joint lot sizing and sequencing with genetic algorithms for scheduling : Evolving the chromosome structure. In *[For93]*, pages 383–389, 1993.

[Lév94] Gérard Lévy. *Algorithmique Combinatoire – Méthodes constructives.* Dunod, 1994. ISBN: 2-1000-2149-4.

[MA94] Zbigniew Michalewicz et Naguib F. Attia. Evolutionary optimization of constrained problems. In *EP'94*. EP Society, 1994.

[Maq96] H. El Maqrini. Production scheduling with metaheuristics in a chemical firm. In *[FUC96]*, pages 356–359, Septembre 1996.

[Mar94] Jonathan G. Maresky. On effective communication in distributed genetic algorithms. Master's thesis, Novembre 1994.

[Mar97] Elena Marchiori. Combining constraint processing and genetic algorithms for constraint satisfaction problems. In *[Bäc97]*, pages 330–337, 1997.

[MB96a] Mihaela Mares et Marc Bourcerie. Modelling sequencing problems with F_p-petri nets. In *[FUC96]*, pages 369–372, Septembre 1996.

[MB96b] Dirk C. Mattfeld et Christian Bierwirth. A search space analysis of the job shop scheduling problem. *Baltzer Journals*, Avril 1996.

[McC97] K. John McConnell. An attempt to determine molecular structure via genetic algorithms. In *[Koz97]*, pages 138–146, 1997.

[MDA94] Frédéric Médioni, Nicolas Durand, et Jean-Marc Alliot. Algorithmes génétiques et programmation linéaire appliqués à la résolution de conflits atc. In *[ALRS94]*, pages 91–98, 1994.

[MDRS96] Zbigniew Michalewicz, Dipankar Dasgupta, Rodolphe G. Le Riche, et Marc Schoenauer. Evolutionary algorithms for constrained engineering problems. *Computers and Industrial Engineering Journal, special issue on Genetic Algorithms and Industrial Engineering*, 30(4):851–870, 1996.

[MdWS91] Bernard Manderick, Mark de Weger, et Piet Spiessens. The genetic algorithm and the structure of the fitness landscape. In *[BB91]*, pages 143–150, 1991.

[MF97] Mark M. Meysenburg et James A. Foster. The quality of pseudo-random number generators and simple genetic algorithm performance. In *[Bäc97]*, pages 276–282, 1997.

[MG92a] S. W. Mahfoud et D. E. Golberg. A genetic algorithm for parallel simulated annealing. In *[MM92]*, pages 301–310, 1992.

[MG92b] Samir W. Mahfoud et David E. Goldberg. Parallel recombinative simulated annealing: A genetic algorithm. Rapport technique IlliGALs report 92002, IlliGALs, department of General Engineering, University of Illinois at Urbana-Champaign, 117 Transportation Building, 104 South Mathews Avenue, Urbana, IL 61801, USA, Juillet 1992.

[MH93a] Melanie Mitchell et John H. Holland. When will a genetic algorithm outperform hill climbing ? In *[For93]*, page 646, 1993. (one page summary).

[MH93b] Toshinori Munakata et David J. Hashier. A genetic algorithm applied to the maximum flow problem. In *[For93]*, pages 488–493, 1993.

[MHL00] Chantal Morley, Jean Hugues, et Bernard Leblanc. *UML pour l'analyse d'un système d'information.* Dunod, 2000. ISBN: 2-100-04826-0.

[Mic92] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, 1992. ISBN: 3-540-55387-8.

[Mic94] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, 2 edition, 1994. ISBN: 3-540-58090-5.

[Mic95] Zbigniew Michalewicz. Genetic algorithms, numerical optimization, and constraints. In *[Esh95]*, pages 151–158, 1995.

[Mic96] Zbigniew Michalewicz. Heuristic methods for evolutionary computation. *Journal of Heuristics*, 1(2):177–206, 1996. ISSN: 1381-1231.

[Mit96] Melanie Mitchell. *An Introduction to Genetic Algorithms.* MIT Press, Cambridge, MA, USA, 1996. ISBN: 0-262-13316-4.

[MK93] Raul San Martin et John P. Knight. Genetic algorithms for optimization of integrated circuits synthesis. In *[For93]*, pages 432–438, 1993.

[ML97] L. Mynard et J.-M. Labat. Combination of oscillating local search and heuristically ordered implicit enumeration. In *[INR97]*, pages 289–292, 1997.

[MM92] R. Männer et B. Manderick, éditeurs. *Parallel Problem Solving from Nature – PPSN II.* Elsevier Science Publishers, Amsterdam, 1992.

[MM98] Thierry Mautor et Philippe Michelon. Mimausa : une nouvelle méthode combinant résolution exacte et recherche locale. In *Premier congrès de la Société Française de Recherche Opérationnelle et Aide à la Décision (ROADF'98)*, page 24, Paris, France, Janvier 1998. (one page abstract).

[MO96] Olivier C. Martin et Steve W. Otto. Combining simulated annealing with local search heuristics. *Annals of Operations Research*, 63:57–75, 1996.

[MPGL93] John A. Miller, Walter D. Potter, Ravi V. Gandham, et Chito N. Lapena. An evaluation of local improvement operators for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 23(5):1340–1993, 1993.

[MPT+99] K. Mesghouni, P. Pesin, D. Trentesaux, S. Hammadi, C. Tahon, et P. Borne. Hybrid approach to decision-making for jobshop scheduling. *Production Planning & Control*, 10(7):690–706, 1999. ISSN: 0953-7287.

[MSB91] Heinz Mühlenbein, M. Schomisch, et J. Born. The parallel genetic algorithm as function optimizer. In *[BB91]*, pages 271–278, 1991.

[MST97] H. M'Silti, A. Schaal, et P. Tolla. Comparison of simulated annealing and genetic algorithm within a hybrid methods for general integer problems. In *[INR97]*, pages 49–50, 1997.

[MT96] H. El Maqrini et J. Teghem. Scheduling general flexible job-shop with several constraints. In *[FUC96]*, pages 352–355, Septembre 1996.

[MTS93] Masaharu Munetomo, Yoshiaki Takai, et Yoshiharu Sato. An efficient migration scheme for subpopulation-based asynchronously parallel genetic algorithms. In *[For93]*, page 649, 1993. (one page summary).

[MV94] András Márkus et József Váncza. Inference and optimization methods for manufacturing process planning. In *[Eib94]*, pages 595–599, 1994.

[NA94] W.P.M. Nuijten et E.H.L. Aarts. Constraint satisfaction for multiple capacited job shop scheduling. In *[Eib94]*, pages 635–639, 1994.

[NDY94] Ryohei Nakano, Yuval Davidor, et Takeshi Yamada. Optimal population size under constant computation cost. In *[DSM94]*, pages 130–138, 1994.

[Ngu97] Khanh V. Nguyen. Improving the crossover operator in genetic algorithms and applications in optimal conference room booking. In *[Koz97]*, pages 178–186, 1997.

[NS96a] Gerhard Niemeyer et Patricia Shiroma. Production scheduling with genetic algorithms and simulation. In *[VERS96]*, pages 930–939, 1996.

[NS96b] Eugeniusz Nowicki et Czeslaw Smutnicki. A fast taboo search algorithm for the job shop problem. *Management Science*, 42(6):797–813, Juin 1996.

[NS97] Eugeniusz Nowicki et Czeslaw Smutnicki. Flow line scheduling by tabu search. In *[INR97]*, page 81, 1997. (one page summary).

[NY91] Ryohei Nakano et Takeshi Yamada. Conventional genetic algorithm for job shop problems. In *[BB91]*, pages 474–479, 1991.

[NY92] Ryohei Nakano et Takeshi Yamada. A genetic algorithm applicable to large-scale job-shop problems. In *[MM92]*, pages 281–290, 1992.

[OD93] David Orvosh et Lawrence Davis. Shall we repair ? genetic algorithms, combinatorial optimization, and feasibility constraints. In *[For93]*, page 650, 1993. (one page summary).

[OD98] Sofiane Oussedik et Daniel Delahaye. Reduction of air traffic congestion by genetic algorithms. In *[EBSS98]*, pages 855–864, Septembre 1998.

[OFM92] S. Oliker, M. Furst, et O. Maimon. A distributed genetic algorithm for neural network design and training. *Complex Systems*, 6(5):459–477, Octobre 1992.

[OK95] Ibrahim H. Osman et James P. Kelly. Meta-heuristics: An overview. In *Proceedings of Meta-Heuristics'95 (MIC'95)*. Kluwer Academic Press, Juillet 1995.

[OL96] Ibrahim H. Osman et Gilbert Laporte. Metaheuristics: A bibliography. *Annals of Operational Research*, 63:513–628, 1996.

[OTT98] Shigeru Obayashi, Shinichi Takahashi, et Yukihiro Takeguchi. Niching and elitist models for mogas. In *[EBSS98]*, pages 260–269, Septembre 1998.

[Pae94] Ben Paechter. Optimising a presentation timetable using evolutionary algorithms. In *[Fog94a]*, pages 264–276, 1994.

[Par92] Jan Paredis. Exploiting constraints as background knowledge for genetic algorithms : a case-study for scheduling. In *[MM92]*, pages 229–238, 1992.

[Par93] Kihong Park. A lower-bound result on the power of genetic algorithms. In *[For93]*, page 651, 1993. (one page summary).

[Par95] Kihong Park. A comparative study of genetic search. In *[Esh95]*, pages 512–519, 1995.

[PD99] Philippe Preux et David Duvivier. Creating gradient to help local search algorithm — application to tabu search for the simple job-shop-scheduling problem. In *Proceedings of the Twelfth European Chapter on Combinatorial Optimization*, Mai 1999.

[Pen94] Bernard Penz. *Constructions agrégatives d'ordonnancements pour des job-shops statiques, dynamiques et réactifs*. PhD dissertation, Université Joseph Fourier, Grenoble I, France, Décembre 1994.

[PG95] Marie-Claude Portmann et Fatima Ghedjati. Affectation et ordonnancement par des algorithmes génétiques, 1995.

[PK94] V. Petridis et S. Kazarlis. Varying quality function in genetic algorithms and the cutting problem, 1994.

[PKS93] H. B. Penfold, U. Kohlmorgen, et H. Schmeck. Deriving application-specific neural nets using a massively parallel genetic algorithm, 1993. submitted to Complex Systems.

[PM98] G.T. Parks et I. Miller. Selective breeding in a multiobjective genetic algorithm. In *[EBSS98]*, pages 250–260, Septembre 1998.

[Por96] Marie-Claude Portmann. Genetic algorithms and scheduling: a state of the art and some propositions. In *[FUC96]*, pages i–xxiv, Septembre 1996.

[PR72] Jean-Marie Proth et André Razafindrakoto. Amélioration d'un ordonnancement dans le cas d'une fabrication de type linéaire. *RAIRO*, 3:17–26, 1972.

[PRCF98] Ben Paechter, R. C. Rankin, Andrew Cumming, et Terence C. Fogarty. Timetabling the classes of an entire university with an evolutionary algorithm. In *[EBSS98]*, pages 865–874, Septembre 1998.

[Pre94a] Philippe Preux. A study of population uniformization in GAs. In *[Eib94]*, 1994.

[Pre94b] Philippe Preux. Étude de l'uniformisation de la population des algorithmes génétiques. In *[ALRS94]*, pages 19–28, 1994.

[Pre95] Philippe Preux. Les algorithmes évolutifs. Rapport technique LIL-1994-01, Laboratoire d'Informatique du Littoral, Calais, France, Octobre 1995.

[Pre98] IEEE Press, éditeur. *Proceedings of ICEC'98-IEEE 5th International Conference on Evolutionary Computation*, Anchorage, Alaska, USA, Mai 1998.

[Pre99] Philippe Preux. *Réflexions sur les systèmes complexes comme outil d'optimisation, leur modélisation et leur simulation.* Thèse d'habilitation à diriger des recherches, Université du Littoral Côte-d'Opale, Calais, France, Janvier 1999.

[PRF97a] Philippe Preux, Denis Robilliard, et Cyril Fonlupt. Fitness landscape and the performance of local search algorithms. In *Proceedings of the Tenth European Chapter on Combinatorial Optimization*, page 54, Mai 1997. (one page summary).

[PRF97b] Philippe Preux, Denis Robilliard, et Cyril Fonlupt. Fitness landscapes of combinatorial problems and the performance of search algorithms. Rapport technique LIL-97-13, Laboratoire d'Informatique du Littoral, Calais, France, Laboratoire d'Informatique du Littoral, Calais, Novembre 1997. (submitted).

[PRF+99] Philippe Preux, Denis Robilliard, Cyril Fonlupt, El-Ghazali Talbi, et Vincent Bachelet. Reaching summits is not wandering, or getting insight into problem landscapes to go higher, faster. Rapport technique LIL-99-5, Laboratoire d'Informatique du Littoral, Calais, Laboratoire d'Informatique du Littoral, Calais, Janvier 1999.

[Pro94] Patrick Prosser. Binary constraint satisfaction problems : Some are harder than others. In *[Eib94]*, pages 95–99, 1994.

[PS82] Christos H. Papadimitriou et Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Inc, Englewood Cliffs, NJ, USA, 1982. ISBN: 0-1315-2462-3.

[PS93] David Powell et Michael M. Skolnick. Using genetic algorithms in engineering design optimization with non-linear constraints. In *[For93]*, pages 424–431, 1993.

[PS96] Jean-Marie Proth et Nathalie Sauer. Continuous approach of scheduling problems based on petri nets. Rapport technique 2822, INRIA Rennes, Mars 1996.

[PT95a] Philippe Preux et El-Ghazali Talbi. Assessing the evolutionary algorithm paradigm to solve hard problems. In *Constraint Processing, workshop on really hard problem solving*, Septembre 1995. An extended version is available as report LIL-1995-04 from the Laboratoire d'Informatique du Littoral (`ftp://ftp-lil/pub/preux/papers/lil-95-4.ps.gz`).

[PT95b] Philippe Preux et El-Ghazali Talbi. Assessing the evolutionary algorithm paradigm to solve hard problems. Rapport technique LIL-1995-04, Laboratoire d'Informatique du Littoral, Calais, France, 1995. (a summary was published in the *Proceedings of the Workshop on Studying and Solving Really Hard Problems, Constraint Processing'95*, Marseille, 1995).

[PvGvL+92] J. Paul, E. von Goldammer, R. van Leendert, D. Picu, J. Stender, E. David, et M. Pfotenhauer. Applications of neural networks and genetic algorithms in the disgnosis of cancer, anorexia nervosa and aids. In *[Ste92]*, pages 187–214, 1992.

[PX95] J. M. Proth et X. Xie. *Les réseaux de Petri pour la conception et la gestion des systèmes de production*. Masson, 1995. ISBN: 2-225-84649-9.

[QP94] Xiafeng Qi et Francesco Palmieri. Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space, part i: Basic properties. *IEEE Transactions on Neural Networks*, 5(1):102–119, Janvier 1994.

[RAARS98] Alvaro Ruiz-Andino, Lourdes Araujo, Jose Ruz, et Fernando Saenz. Parallel evolutionary optimisation with constraint propagation. In *[EBSS98]*, pages 270–279, Septembre 1998.

[RAE96] F. Riane, A. Artiba, et S. E. Elmaghraby. Sequencing on hybrid two-stages flowshop to minimize makespan. In *[FUC96]*, pages 93.1–93.8, Septembre 1996.

[Ram97] E. Ramat. *Modélisation et planification de projets complexes à contraintes de ressources : le modèle RAIH*. PhD thesis, Université de Tours, École d'Ingénieurs en Informatique pour l'Industrie, Tours, France, Juin 1997.

[RAT97] Felipe Rodriguez-Acosta et Néstor V. Torres. Optimization of biotechnological processes by evolutionary algorithms. application to the maximization of the rate production of ethanol, glycerol and carbohydrate production by *saccharomyces cerevisiae*. In *Proceedings of the Tenth European Chapter on Combinatorial Optimization*, page 62, Mai 1997. (one page summary).

[Raw91] Gregory J. E. Rawlins, éditeur. *Proceedings of the Workshop on the Foundations of Genetic Algorithms*, Bloomington, Indiana, USA, 1991. Morgan Kaufmann, San Mateo, CA, USA.

[RC94] Peter Ross et Dave Corne. Applications of genetic algorithms. In Terry Fogarty, éditeur, *Special issue of the AISB Quarterly on Evolutionary Computation*, 1994. Paper No. 94-007.

[RCF94a] Peter Ross, Dave Corne, et Hsiao-Lan Fang. Improving evolutionary timetabling with delta evaluation and directed mutation. In *[DSM94]*, pages 556–565, 1994.

[RCF94b] Peter Ross, Dave Corne, et Hsiao-Lan Fang. Successful lecture timetabling with evolutionary algorithms. In *[Eib94]*, pages 1–12, 1994.

[REA95] Helge Rosé, Werner Ebeling, et Torsten Asselmeyer. The density of states - a measure of the difficulty of optimisation problems. In *[Esh95]*, pages 208–217, 1995.

[Ree95] Colin R. Reeves, éditeur. *Modern Heuristic Techniques for Combinatorial Problems*. Advance Topics in Computer Science. Mc Graw-Hill, 1995. ISBN: 0-07-709239-2.

[ren96] *RenPar'8 – 8èmes Rencontres Francophones du Parallélisme*, Bordeaux, France, Mai 1996.

[RFRT98] Olivier Roux, Cyril Fonlupt, Denis Robilliard, et El-Ghazali Talbi. Antabu. In *ANTS*, Bruxelles, Belgique, Octobre 1998.

[RH96] Bert De Reyck et Willy Herroelen. A branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations. In *[FUC96]*, pages 344–347, Septembre 1996.

[RH98] Peter Ross et Emma Hart. An adaptive mutation scheme for a penalty-based graph-colouring ga. In *[EBSS98]*, pages 795–802, Septembre 1998.

[RHCC97] Kwang Ryel Ryu, Junha Hwang, Hyung Rim Choi, et Kyu Kab Cho. A genetic algorithm hybrid for hierarchical reactive scheduling. In *[Bäc97]*, pages 497–504, 1997.

[Rio92] Rick Riolo. La recherche de l'absolu. *Pour la Science*, 179:101–103, Septembre 1992.

[RK98] Günther R. Raidl et Gabriele Kodydek. Genetic algorithms for the multiple container packing problem. In *[EBSS98]*, pages 875–884, Septembre 1998.

[RKLH95] Rodolphe G. Le Riche, Catherine Knopf-Lenoir, et Raphael T. Haftka. A segregated genetic algorithm for constrained structural optimization. In *[Esh95]*, pages 558–565, 1995.

[RLTP96] E. Ramat, C. Lente, C. Tacquard, et C. Proust. The RAIH model: application on product realization process example. In *[FUC96]*, pages 178–183, Septembre 1996.

[RO93] Adel Torkaman Rahmani et Norihiko Ono. A genetic algorithm for channel routing problem. In *[For93]*, pages 494–498, 1993.

[Roc98] Sophie Rochet. *Convergence des algorithmes génétiques : modèles stochastiques et épistasie*. PhD thesis, Université de Provence, Janvier 1998.

[Rod94] Robert Rodošek. Combining constraint network and causal theory to solve scheduling problems from a csp perspective. In *[Eib94]*, pages 630–634, 1994.

[Rou98] Olivier Roux. Modèles parallèles pour les colonies de fourmis. Master's thesis, Université des Sciences et Technologies de Lille 1, Villeneuve d'Ascq, Juillet 1998. DEA thesis, Laboratoire d'Informatique Fondamentale de Lille.

[RP96] R. Roy et I. C. Parmee. Adaptive restricted tournament selection and a local hill climbing for the identification of multiple "good" design solutions, Mars 1996. Second Online Workshop on Evolutionary Computation.

[RPLH89] Jon T. Richardson, Mark R. Palmer, Gunar Liepins, et Mike Hilliard. Some guidelines for genetic algorithms with penalty functions. In *[Sch89]*, pages 190–197, 1989.

[RSORS96] V.J. Rayward-Smith, I.H. Osman, C.R. Reeves, et G.D. Smith, éditeurs. *Modern Heuristic Search Methods*. J. Wiley & Sons Ldt., 1996. ISBN: 0-471-96280-5.

[Rud94a] Günter Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5(1):96–101, Janvier 1994.

[Rud94b] Günter Rudolph. An evolutionary algorithm for integer programming. In *[DSM94]*, pages 139–148, 1994.

[Rut89] Rob A. Rutenbar. Simulated annealing algorithms: An overview. *IEEE Circuits and Devices*, 5(1):19–26, Janvier 1989.

[RVLS97] E. Ramat, G. Venturini, C. Lente, et M. Slimane. Solving the multiple resource constrained project scheduling problem with a hybrid genetic algorithm. In *[Bäc97]*, pages 489–496, 1997.

[RVSK97] S. Rochet, G. Venturini, M. Slimane, et E. M. El Kharoubi. A critical and empirical study of epistasis measures for predicting GA performances: a summary. In *[HLR+97]*, pages 331–341, 1997.

[SAS92] Joachim Stender, Thomas R. Addis, et Susan Ella Spenceley. Principle-based engineeing and economic modelling. In *[Ste92]*, pages 117–128, 1992.

[SAW99] K. Steinhöfel, A. Albrecht, et C.K. Wong. Two simulated annealing-based heuristics for the job shop scheduling problem. *European Journal of Operational Research*, 118:524–548, 1999.

[Sch85] J. David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *[Gre85]*, pages 93–100, 1985.

[Sch89] J.D. Schaffer, éditeur. *Proceedings of the Third International Conference on Genetic Algorithms*, Bloomington, IN, USA, 1989. Morgan Kaufmann, San Mateo, CA, USA. ISBN: 1-55860-066-3.

[Sch90] Jonathan Schull. The view from the adaptive landscape. In *[SM91]*, pages 415–427, 1990.

[SD90] William M. Spears et Kenneth A. De Jong. Using neural networks and genetic algorithms as heuristics for NP-complete problems. In *Proceedings of International Joint Conference On Neural Networks*, 1990.

[SD96] Arno Sprecher et Andreas Drexl. Truncated branch-and-bound methods for solving resource-constrained project scheduling problems. In *[FUC96]*, pages 8–11, Septembre 1996.

[SDMW96] T. Starkweather, S. Mc Daniel, K. Mathias, et D. Whitley. A comparison of genetic sequencing operators. In *[BB91]*, pages 69–76, 1996.

[SERW95] F. Schweitzer, W. Ebeling, H. Rosé, et O. Weiss. Network optimization using evolutionary strategies. In *[Esh95]*, pages 940–949, 1995.

[SF96] K.J. Shaw et P.J. Fleming. Initial study of multi-objective genetic algorithms for scheduling the production of chilled ready meals. In *Proceedings of the Second International Mendel conference on Genetic Algorithms – Mendel'96*, Juin 1996.

[SF97] K.J. Shaw et P.J. Fleming. Which line is it anyway ? use of rules and preferences for schedule builders in genetic algorithms production scheduling. In *Proceedings of the AISB Workshop on Evolutionary Computation*, Avril 1997.

[SG80] Monique Spielberg-Guignard. *Contribution à l'étude de l'optimisation en nombres entiers et de l'optimalité en programmation mathématique.* PhD thesis, Université des Sciences et Technologies de Lille I, Lille, Septembre 1980.

[SG87] Jung Y. Suh et Dirk Van Gucht. Incorporating heuristic information into genetic search. In *[Gre87a]*, pages 100–107, 1987.

[SG97a] P. Soriano et M. Gendreau. The max-min ant system and local search for combinatorial optimization problems : Towards adaptive tools for global optimization. In *[INR97]*, pages 191–193, 1997.

[SG97b] Patrick Soriano et Michel Gendreau. Parallel tabu search with a variable number of threads : An application to the mcp. In *[INR97]*, page 143, 1997. (one page summary).

[SH87] Bart Selman et Graeme Hirst. Parsing as an energy minimization problem. In *[Dav87]*, pages 141–154. Morgan Kaufmann, San Mateo, CA, USA, 1987.

[Sha98] J. L. Shapiro. Does data-model co-evolution improve generalization performance of evolving learners ? In *[EBSS98]*, pages 540–549, Septembre 1998.

[Sin98] Mark C. Sinclair. Operator-probability adaptation in a genetic-algorithm/heuristic hybrid for optical network wavelength allocation. In *[Pre98]*, pages 840–845, 1998.

[SK92] S. Schulze-Kremer. Genetic algorithms for protein tertiary structure prediction. In *[Ste92]*, pages 129–149, 1992.

[SL90] Jung-Yul Suh et Chan-Do Lee. Operator-oriented genetic algorithm and its application to sliding block puzzle problem. In *[SM91]*, pages 98–103, 1990.

[SM91] H-P. Schwefel et R. Männer, éditeurs. *Parallel Problem Solving from Nature – PPSN I.* Springer-Verlag, Berlin, Octobre 1991. Lecture Notes in Computer Science, vol. 496.

[SM95] Timothy J. Stanley et Trevor Mudge. A parallel genetic algorithm for multiobjective microprocessor design. In *[Esh95]*, pages 597–604, 1995.

[Soa94] C. Soares. Evolutionary computation for the job-shop scheduling problem. Rapport technique UU-CS-1994-52, Utrecht University, The Netherlands, Décembre 1994.

[SP91] Gilbert Syswerda et Jeff Palmucci. The application of genetic algorithms to resource scheduling. In *[BB91]*, pages 502–508, 1991.

[Spe91] William M. Spears. On the virtues of parameterized uniform crossover. In *[BB91]*, pages 230–236, 1991.

[SS83] Milton L. Smith et Abraham Seidmann. Due date selection procedures for job-shop simulation. *Computers and Industrial Engineering Journal*, 7(3):199–207, 1983.

[SS89] W. Siedlecki et J. Sklansky. Constrained genetic optimization via dynamic reward-penalty balancing and its use in pattern recognition. In *[Sch89]*, pages 141–150, 1989.

[SS94] Robert E. Smith et Ellen Smuda. Adaptatively resizing populations: Algorithm, analysis, and first results. *Evolutionary Computation*, 2, 1994. ISSN: 1063-6560.

[SSR94] Michèle Sebag, Marc Schoenauer, et Caroline Ravisé. Évolution darwinienne ou Évolution civilisée ? In *[ALRS94]*, pages 77–84, 1994.

[SSW93] David B. Shmoys, Clifford Stein, et Joel Wein. Improved approximation algorithms for shop scheduling problems. Rapport technique TR-921, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853, USA, 1993. A Preliminary version of this paper appeared in the proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms, January 1991.

[ST93] Alice E. Smith et David M. Tate. Genetic optimization using a penalty function. In *[For93]*, pages 499–505, 1993.

[ST94] Yu. N. Sotskov et V. S. Tanaev. Scheduling theory and practice : Minsk group results. *Intelligent Systems Engineering*, Spring 1994.

[Ste92] Joachim Stender, éditeur. *Parallel Genetic Algorithms: Theory and Applications*. IOS Press, Amsterdam, 1992.

[Sun95] Xiao-Chao Sun. A new branch and bound method for integer linear programming. In *International Workshop on Combinatorics and Computer Science (CCS'95)*, pages 101–102, Brest, France, Juillet 1995.

[SV95] Volker Schnecke et Olivier Vornberger. An adaptative parallel genetic algorithm for vlsi-layout optimization. In *[Esh95]*, pages 859–868, 1995.

[SV97] Lutz Sondergeld et Stefan Voss. Cooperative intelligent agent systems. In *[INR97]*, pages 169–172, 1997.

[SW87] David J. Sirag et Paul T. Weisser. Toward a unified thermodynamic genetic operator. In *[Gre87a]*, pages 116–122, 1987.

[SX93] Marc Schoenauer et Spyros Xanthakis. Constrained ga optimization. In *[For93]*, pages 573–580, 1993.

[Tai93a] E. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64:278–285, 1993.

[Tai93b] E. Taillard. *Recherches itératives dirigées parallèles*. PhD dissertation, Ecole Polytechnique Fédérale de Lausanne, 1993.

[Tal98] El-Ghazali Talbi. A taxonomy of hybrid meta-heuristics. Rapport technique AS-183, Laboratoire d'Informatique Fondamentale de Lille, Université de Lille I, 59655 Villeneuve d'Ascq Cedex, France, Mai 1998.

[Tan89] Reiko Tanese. Distributed genetic algorithms. In *[Sch89]*, pages 434–439, 1989.

[TBAM] El-Ghazali Talbi, Pierre Bessière, Juan-Manuel Ahuactzin, et Emmanuel Mazer. Algorithmes génétiques parallèles et leurs applications. (soumis à TSI).

[TCM95] R. Tadei, F. Della Croce, et G. Menga. Advanced search techniques for the job shop problem: a comparison. *Operations Research*, 29(2):179–194, 1995.

[TEM95] E. G. Talbi, A. Elleuch, et T. Muntean. Une approche hybride pour le placement de processus. In *Journée de Recherche sur le placement dynamique et la répartition de charge : application aux systèmes parallèles et distribués*, pages 51–53, Paris, France, Mai 1995.

[TF93] Shigeyoshi Tsutsui et Yoshiji Fujimoto. Forking genetic algorithm with blocking and shrinking modes (fGA). In *[For93]*, pages 206–213, 1993.

[TG97] E. D. Taillard et L. M. Gambardella. An ant approach for structured quadratic assignment problems. In *[INR97]*, pages 217–222, 1997.

[THG97] El-Ghazali Talbi, Zouhir Hafidi, et Jean-Marc Geib. Parallel adaptive tabu search for large optimization problems. In *[INR97]*, pages 137–142, 1997.

[THG98] El-Ghazali Talbi, Zouhir Hafidi, et Jean-Marc Geib. A parallel adaptive tabu search approach. *Parallel Computing*, 24(14):2003–2019, Décembre 1998.

[TMK97] K. S. Tang, K. F. Man, et K. T. Ko. Wireless LAN design using hierarchical genetic algorithm. In *[Bäc97]*, pages 629–635, 1997.

[TMS94] E.-G. Talbi, T. Muntean, et I. Samarandache. Hybridation des algorithmes génétiques avec la recherche tabou. In *[ALRS94]*, pages 85–90, 1994.

[TN92] Hisashi Tamaki et Yoshikazu Nishikawa. A paralleled genetic algorithm based on a neighborhood model and its application to the jobshop scheduling. In *[MM92]*, pages 573–582, 1992.

[Tom96] M. Tomassini. Evolutionary algorithms. In E. Sanchez et M. Tomassini, éditeurs, *Towards Evolvable Hardware, volume 1062 of Lecture Notes in Computer Science*, pages 19–47. Springer-Verlag, Berlin, 1996.

[TR98] Andrew Tuson et Peter Ross. Adapting operator settings in genetic algorithms. *Evolutionary Computation*, 6(2), 1998. ISSN: 1063-6560.

[TRFR99] El-Ghazali Talbi, Olivier Roux, Cyril Fonlupt, et Denis Robilliard. Parallel ant colonies for combinatorial optimization problems. In José Rolim et al., éditeur, *13th International Parallel Processing Symposium, volume 1586 of Lecture Notes in Computer Science*, pages 239–247. Springer-Verlag, Berlin, San Juan, Puerto Rico, USA, Avril 1999.

[TS97] David S. Todd et Pratyush Sen. A multiple criteria genetic algorithm for containership loading. In *[Bäc97]*, pages 674–681, 1997.

[TVG93] Sam R. Thangiah, Rajini Vinayagamoorthy, et Ananda V. Gubbi. Vehicle routing with time deadlines using genetic and local algorithms. In *[For93]*, pages 506–513, 1993.

[UBKM93] S. Uckun, S. Bagchi, K. Kawamura, et Y. Miyabe. Managing genetic search in job shop scheduling. In *IEEE Expert*, pages 15–24, Octobre 1993.

[VAL92] R.J.M. Vaessens, E.H.L. Aarts, et J.K. Lenstra. A local search template. In *[MM92]*, pages 65–74, 1992.

[VAL96] R.J.M. Vaessens, E.H.L. Aarts, et J.K. Lenstra. Job shop scheduling by local search, 1996.

[Van78] Michel Vanbreugel. *Un algorithme général pour l'optimisation non linéaire.* PhD thesis, Université des Sciences et Technologies de Lille I, Lille, Novembre 1978.

[Vas97] Vesselin K. Vassilev. An information measure of landscapes. In *[Bäc97]*, pages 49–56, 1997.

[VB66] T. E. Vollmann et E. S. Buffa. The facility layout problem in perspective. *Management Science*, 12(10):450–468, 1966.

[VDDP96] A. Vignier, D. Dardilhac, D. Dezalay, et C. Proust. An optimal approach to solve a k-stage hybrid flowshop. In *[FUC96]*, pages 315–319, Septembre 1996.

[Ven97] Gilles Venturini. *Apport des algorithmes génétiques à l'apprentissage et à l'optimisation.* Thèse d'habilitation à diriger des recherches, Université François Rabelais, Tours, France, Décembre 1997.

[VERS96] H-M. Voight, W. Ebeling, I. Rechenberg, et H-P. Schwefel, éditeurs. *Parallel Problem Solving from Nature – PPSN IV*, Berlin, Germany, 1996. Springer-Verlag, Berlin. Lecture Notes in Computer Science, vol. 1141.

[VFM95] Rémy Viennet, Christian Fonteix, et Ivan Marc. Optimisation multicritère à l'aide d'un algorithme génétique diploïde. In *Actes de Évolution Artificielle'95*, pages 143–150, Brest, France, Septembre 1995.

[VHS98] Christine Valenzuela, Steve Hurley, et Derek Smith. A permutation based genetic algorithm for minimum span frequency assignment. In *[EBSS98]*, pages 907–916, Septembre 1998.

[vKHHK95] C.H.M. van Kemenade, C.F.W. Hendriks, H.H. Hesselink, et J.N. Kok. Evolutionary computation in air traffic control planning. In *[Esh95]*, pages 611–616, 1995.

[Vli98] John Vlissides. *Pattern Hatching - Design Patterns Applied.* Addison-Wesley, 1998. ISBN: 0-201-43293-5.

[vLM90] Gregor von Laszewski et Heinz Mühlenbein. Partitioning a graph with a parallel genetic algorithm. In *[SM91]*, pages 165–169, 1990.

[VRUC97] Manuel Valenzuela-Rendon et Eduardo Uresti-Charre. A non-generational genetic algorithm for multiobjective optimization. In *[Bäc97]*, pages 658–665, 1997.

[vWKvdBvK94] M.C. van Wezel, J.N. Kok, J. van den Berg, et W. van Kampen. Genetic improvement of railway timetables. In *[DSM94]*, pages 566–575, 1994.

[WBV99] Jo Wyns, Hendrik Van Brussel, et Paul Valckenaers. Design pattern for deadlock handling in holonic manufacturing systems. *Production Planning & Control*, 10(7):616–626, 1999. ISSN: 0953-7287.

[WFMS94] J. D. Walker, P. E. File, C. J. Miller, et W. B. Samson. A hybrid genetic algorithm application to a genetic sequencing problem. In *IEE Colloquium on Molecular Bioinformatics*, 1994.

[Whi89] Darrell Whitley. The Genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In *[Sch89]*, pages 116–121, 1989.

[Whi93a] Darrell Whitley. Cellular genetic algorithms. In *[For93]*, page 658, 1993. (one page summary).

[Whi93b] Darrell Whitley. A genetic algorithm tutorial. Rapport technique CS-93-108, Colorado State University; Department of Computer Science, Fort Collins, CO 80523, USA, Août 1993.

[Whi93c] L. Darrell Whitley, éditeur. *Proceedings of the Workshop on Foundations of Genetic Algorithms*, Vail, Colorado, USA, 1993. Morgan Kaufmann, San Mateo, CA, USA.

[WM95] David H. Wolpert et William G. Macready. No free lunch theorems for search. Rapport technique SFI-TR-95-02-010, The Santa Fe Institute, Santa Fe, NM, Novembre 1995.

[WM97] David H. Wolpert et William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, Avril 1997.

[WS94] Zhiming Wu et Marc Schoenauer. Éléments finis adaptatifs : Couplage entre programmation génétique et méthode des Éléments finis. In *[ALRS94]*, pages 125–131, 1994.

[WSF89] Darrell Whitley, Thimothy Starkweather, et D'Ann Fuquay. Scheduling problems and traveling salesmen: The genetic edge recombination operator. In *[Sch89]*, pages 133–140, 1989.

[WSL95] Avinash M. Waikar, Bhaba R. Sarker, et Anil M. Lal. A comparative study of some priority dispatching rules under different shop loads. *Production Planning & Control*, 6(4):301–310, 1995. ISSN: 0953-7287.

[WSS91] Darrell Whitley, Timothy Starkweather, et Daniel Shaner. The traveling salesman and sequence scheduling: Quality solutions using genetic edge recombination. In Lawrence Davis, éditeur, *[Dav91b]*, chapitre 22, pages 350–372. Van Nostrand Reinhold, New York, USA, 1991.

[WV94] Darrell Whitley et Michael D. Vose, éditeurs. *Proceedings of the Workshop on Foundations of Genetic Algorithms*, Estes Park, Corado, USA, 1994. Morgan Kaufmann, San Mateo, CA, USA.

[WY94] Darrell Whitley et Nam-Wook Yoo. Modeling simple genetic algorithms for permutation problems. In *[WV94]*, 1994.

[YB98] Tina Yu et Peter Bentley. Methods to evolve legal phenotypes. In *[EBSS98]*, pages 280–291, Septembre 1998.

[YGMTH94] Peng Ye, Derrick Glass, Michael Mc-Tear, et John G. Hughes. Job cost and constraint relaxation for scheduling problem solving in the CLP paradigm. In *[Eib94]*, pages 640–644, 1994.

[YN96] Takeshi Yamada et Ryohei Nakano. Scheduling by genetic local search with multi-step crossover. In *[VERS96]*, pages 960–969, 1996.

[YR97] Takeshi Yamada et Colin R. Reeves. Permutation flowshop scheduling by genetic local search. In *Proceedings of the second IEE/IEEE Int. Conference on Genetic ALgorithms in Engineering Systems: Innovations and Applications (GALESIA'97)*, Glasgow, UK, 1997.

[Yun97] Yeogirl Yun. Finding an optimal blackjack strategy using genetic algorithm. In *[Koz97]*, pages 226–237, 1997.

[Yur94] Deniz Yuret. From genetic algorithms to efficient optimization. Master's thesis, MIT, Cambridge, MA, USA, Mai 1994.

[Özd96] Linet Özdamar. Capacitated family disaggregation in hierarchical production planning. In *[FUC96]*, pages 86–89, Septembre 1996.

[ZK] Bernard P. Zeigler et Jinwoo Kim. Characteristics of an asynchronous parallel genetic algorithm.

[ZK93] Bernard P. Zeigler et Jinwoo Kim. Asynchronous genetic algorihms on parallel computers. In *[For93]*, page 660, 1993. (one page summary).

[ZT98] Eckart Zitzler et Lothar Thiele. Multiobjective optimization using evolutionary algorithms – a comparative case study. In *[EBSS98]*, pages 292–301, Septembre 1998.

Notations

AIRL(f)	Algorithme Itératif de Recherche Locale guidé par la fonction coût f.
$\mathcal{E}$	Espace de recherche.
$\|\mathcal{E}\|$	Cardinal de l'espace de recherche.
f	Fonction coût.
$\overline{f}$	Coût moyen (sur la population, ...).
C	Critère principal.
C'	Critère secondaire.
$\phi_\sharp(C')$	Pouvoir discriminant du critère C'.
$\varphi_\sharp(C')$	Pouvoir discriminant moyen du critère C'.
J	Nombre de produits (jobs) de l'instance de JSP considérée.
M	Nombre de machines de l'instance de JSP considérée.
$\mathcal{J}$	Ensemble des J produits (jobs) de l'instance de JSP considérée ($J = \|\mathcal{J}\|$).
$\mathcal{M}$	Ensemble des M machines de l'instance de JSP considérée ($M = \|\mathcal{M}\|$).
Ω	Ensemble des $J \times M$ opérations de l'instance de JSP considérée ($J \times M = \|\Omega\|$).
x	Un ordonnancement de l'instance de JSP considérée.
M_m	Une machine de l'instance de JSP considérée.
J_j	Un produit (job) de l'instance de JSP considérée.
O_{ji}	$i^{\text{ème}}$ opération du produit J_j.
O_j^m	Opération du produit J_j exécutée sur la machine M_m.
$d(O_{ji})$	Durée de l'opération O_{ji}.
u	Durée maximale des opérations de l'instance considérée.
$DD(O_{ji})$	Date de début de l'opération O_{ji}.
$DF(O_{ji}) = C(O_{ji})$	Date de fin de l'opération O_{ji}.
$DDT(O_{ji})$	Date de début au plus tôt de l'opération O_{ji}.
$DFT(O_{ji})$	Date de fin au plus tard de l'opération O_{ji}.
$EC(O_{ji})$	Date de fin au plus tôt de l'opération O_{ji}.
$E^m(x)$	Date de fin de la dernière opération réalisée sur la machine M_m selon l'ordonnancement x.
$E_j(x)$	Date de fin de la dernière opération du produit J_j selon l'ordonnancement x.
$C_{\max}(x)$	Makespan de l'ordonnancement x.
$C^*_{\max}$	Makespan d'un ordonnancement optimal.
$C_{\text{op}}(x)$	Nombre d'opérations critiques dans l'ordonnancement x.
$\mathcal{O}$	Un opérateur.
$V_{\mathcal{O}}(s)$	Voisinage de la solution courante s obtenu par application de l'opérateur $\mathcal{O}$ à s.
$\|V_{\mathcal{O}}(s)\|$	Taille du voisinage engendré par l'opérateur $\mathcal{O}$ à partir d'une solution s.
$\mathcal{P}$	Paysage adaptatif.
$\mathcal{G}$	Un graphe.

Glossaire

Algorithmes Itératifs de Recherche Locale

Voisin ..... Solution obtenue après une application de l'opérateur à la solution courante.
Fonction coût ..... Mesure la qualité de la solution courante en tant que solution au problème à résoudre.
Évaluation ..... Application de la fonction coût à la solution courante.

Algorithmes Évolutifs

Allèle ..... Valeur d'un gène (entier, réel, permutation d'entiers, ...).
Fonction d'évaluation ... Mesure l'adaptation d'un individu au problème à résoudre.
Chromosome ..... Ensemble de gènes codés selon une structure de données (tableau, scalaire, ...).
Crossover ..... Opérateur de recombinaison.
Diversité ..... Mesure de l'hétérogénéité de la population.
Écart entre génération .. Fraction de la population remplacée d'une génération à la suivante.
Évaluation ..... Application de la fonction d'évaluation à un individu.
Fitness ..... Degré d'adaptation (d'un individu au problème à résoudre).
Gène ..... Partie élémentaire *indivisible* d'un chromosome.
Génération ..... Phase complète de l'algorithme génétique (reproduction, évaluation, sélection).
Génotype ..... Structure d'un individu.
Individu ..... Solution potentielle du problème traité, constituée d'un ou plusieurs chromosomes.
Locus ..... Position dans un chromosome.
Opérateur génétique .... Opérateur agissant sur un ou plusieurs individu(s) ; par exemple : mutation, crossover.
Phénotype ..... Interprétation du génotype en tant que structure de données (*décodage du génotype*).
Population ..... Groupe d'individus sur lequel agissent les opérateurs de l'AE.

Recherche Tabou

Aspirant ..... Voisin de la solution courante qui vérifie le critère d'aspiration.
Critère d'aspiration ..... Critère permettant de lever l'interdiction d'un mouvement tabou si ce dernier conduit à une solution de meilleur coût.
Liste Tabou ..... Structure de données mémorisant les mouvements tabous.
Mouvement Tabou ..... Mouvement interdit pour un certain nombre d'itérations (sauf s'il conduit à une solution de meilleur coût).

Recuit Simulé

Énergie ..... Coût de la solution courante.
État ..... Solution potentielle du problème traité.
Température ..... Principal paramètre du recuit simulé.

Abréviations

ACP ..... Analyse en Composantes Principales.
AE ..... Algorithme Évolutif.
AG ..... Algorithme Génétique.
AIRL ..... Algorithme Itératif de Recherche Locale.
AIRLD ..... Algorithme Itératif de Recherche Locale Déterministe.
AIRLND ..... Algorithme Itératif de Recherche Locale Non Déterministe.
AIRLNDM ..... Algorithme Itératif de Recherche Locale Non Déterministe intégrant une Mémoire des voisins explorés.
AIRLNDP ..... Algorithme Itératif de Recherche Locale Non Déterministe intégrant un mécanisme de déplacement sur les Plateaux.
AIRLNDPM ..... Algorithme Itératif de Recherche Locale Non Déterministe intégrant un mécanisme de déplacement sur les Plateaux et une Mémoire des voisins explorés.
CREGI ..... Centre de Recherches et d'Études en Gestion Industrielle.
DDT ..... Date de Début au plus Tôt.
DFT ..... Date de Fin au plus Tard.
E3I ..... École d'Ingénieurs en Informatique pour l'Industrie.
FUCAM ..... Facultés Universitaires CAtholiques de Mons.
JSP ..... *Jobshop Scheduling Problem* – Problème d'ordonnancement de type *jobshop*.
LAMIH ..... Laboratoire d'Automatique et de Mécanique Industrielles et Humaines.
LIFL ..... Laboratoire d'Informatique Fondamentale de Lille.
LIL ..... Laboratoire d'Informatique du Littoral.
MESC ..... Modélisation, Évolution et Simulation des Systèmes Complexes.
MESC est l'un des trois axes de recherche du Laboratoire d'Informatique du Littoral.
NFL ..... *No Free Lunch*. Théorème NFL, voir section 2.3.2 page 18.
PERFORM ..... *Parallel gEnetic algoRithms FOR optiMization* – PERFORM est un groupe de travail qui rassemble des chercheurs du Laboratoire d'Informatique Fondamentale de Lille (LIFL) et du Laboratoire d'Informatique du Littoral (LIL). Il est possible d'accéder à ce groupe de travail sur Internet via l'adresse `http://www.lifl.fr/~talbi/perform.html`.
PERT ..... *Program Evaluation and Research Technique*
Méthode de modélisation et de résolution de problèmes d'ordonnancement.
PLNE ..... Programmation Linéaire en Nombres Entiers.
PMX ..... Crossover PMX (dédié au TSP).
PSE ..... Procédure par Séparation et Évaluation.
QAP ..... *Quadratic Assignment Problem* – Problème d'affectation quadratique.
RAIH ..... Réseau d'Activités Incertaines Hiérarchisées.
SAT ..... Problème de SATisfaisabilité.
SUS ..... Sélection de type échantillonnage stochastique du reste.
TSP ..... *Traveling Salesman Problem* – Problème du voyageur de commerce.
ULCO ..... Université du Littoral, Côte-d'Opale.
UML ..... *Unified Modeling Language* – Langage de modélisation résultant de l'unification des méthodes de modélisation orientées objet Booch, OOSE et OMT.

Définitions, propriétés et théorèmes

Tableau D.1: Définitions, propriétés et théorèmes

Table des figures

Table des tableaux

Index

D

E

F

G

H

I

J

L

M

N

Q

R

S

T

U

V

Résumé

Dans ce mémoire, nous étudions les méthodes itératives de recherche dans le cadre de la résolution du problème d'ordonnancement de type *jobshop*. Plus que les performances en elles-mêmes, nous nous intéressons tout particulièrement à la compréhension du fonctionnement des méthodes de résolution ainsi qu'à l'analyse de l'influence de la coopération de plusieurs méthodes de recherche sur la qualité des solutions engendrées.

Dans un premier temps, nous évaluons l'apport de critères secondaires intégrés dans la fonction coût. Nous utilisons des algorithmes itératifs de recherche pour étudier l'impact de l'intégration de ces critères sur le paysage adaptatif ainsi que sur la qualité des ordonnancements engendrés.

Nous proposons ensuite quelques améliorations du schéma d'application des opérateurs dans les algorithmes génétiques.

Finalement, nous étudions quelques modèles d'hybridation des méta-heuristiques basés sur la recherche tabou et les algorithmes évolutifs.

Title

Hybridization of meta-heuristics, application to the jobshop scheduling problem

Abstract

In this thesis, iterative search methods are studied and applied to the jobshop scheduling problem. Instead of trying to reach high performance, we focus our work towards a better understanding of search algorithms and their hybridization. In particular, we study the impact of hybridization on the quality of final solutions.

First part of our work consists in evaluating the impact of combining several criteria in the objective function. We have used several local search algorithms in order to gather statistical informations about the resulting structures of fitness landscape. We have also studied the impact on performance.

Then, the study of the interactions between operators in our GA leads us to define some improvements of the scheme of application of genetic-operators.

Finally, we study several schemes of hybridization based on tabu-search and evolutionary algorithms.

Discipline : informatique

Mots-clefs : méta-heuristiques, hybridation, algorithmes évolutifs, recherche tabou, ordonnancement, *jobshop*

Laboratoire : Laboratoire d'Informatique du Littoral – U.L.C.O. – BP719 62228 Calais