Différentes approches logicielles pour la résolution des problèmes combinatoires en temps réel

Bertrand Jullien

UNIVERSITE DES SCIENCES SOCIALES DE TOULOUSE

DIFFERENTES APPROCHES LOGICIELLES POUR LA RESOLUTION DES PROBLEMES COMBINATOIRES EN TEMPS REEL

THESE

Pour le Doctorat de 3e Cycle

Spécialité : Mathématiques appliquées à l'Economie

présentée et soutenue le 16 Avril 1976

par

Bertrand JULLIEN

MEMBRES DU JURY

Président : M. BAZERQUE

Suffragants : MM. DEVILLEBICHOT

MAHL

MOREAUX

UNIVERSITE DES SCIENCES SOCIALES DE TOULOUSE

DIFFERENTES APPROCHES LOGICIELLES POUR LA RESOLUTION DES PROBLEMES COMBINATOIRES EN TEMPS REEL

THESE

Pour le Doctorat de 3^{e} Cycle

Spécialité : Mathématiques appliquées à l'Economie

présentée et soutenue le 16 Avril 1976

par

Bertrand JULLIEN

MEMBRES DU JURY

Président : M. BAZERQUE

Suffragants : MM. DEVILLEBICHOT

MAHL

MOREAUX

UNIVERSITE des SCIENCES SOCIALES de TOULOUSE

PERSONNEL de l'UNIVERSITE

Année Universitaire 1975-1976

HONORARIAT

MM. MAURY, CH. L.H.	Professeur honoraire, doyen honoraire
BARRERE A.	Professeur honoraire, ex-doyen de la Faculté de Droit de PARIS
JAMES, Ch. L.H.	Professeur honoraire , Membre de l'Institut
LASSEGUE,	Professeur à l'Université de PARIS I
RAYNAUD,	Professeur à l'Université de PARIS II
VEDEL, Com. L.H.	Professeur à l'Université de PARIS II
COUZINET, O.L.H.	Professeur honoraire
ROUSSIER, Ch. L.H.	Professeur honoraire

PROFESSEURS, MAITRES de CONFERENCES AGREGES, MAITRES de CONFERENCES, CHARGES de COURS, MAITRES-ASSISTANTS.

PROFESSEURS

MM. PALLARD,	Professeur de Droit Commercial Président de l'Université
HEBRAUD, Ch.L.H.	Professeur de Droit Civil
OURLIAC, O. L.H.	Professeur d'Histoire des Institutions Membre de l'Institut
DAUVILLIER, Ch. L.H.	Professeur de Droit Romain
VIGREUX, Ch. L.H.	Professeur d'Economie Politique
CLUSEAU, Ch. L.H.	Professeur d'Economie Politique Vice-Président de l'Université
LETINIER,	Professeur d'Econométrie
BOYER,	Professeur de Droit Civil
MERLE,	Professeur de Droit Criminel
MONTANE de la ROQUE,	Professeur de Droit Constitutionnel
MARTIN de la MOUTTE,	Professeur de Droit Civil
DUPEYROUX O.	Professeur de Droit Administratif
VELLAS,	Professeur de Droit International Public
VINCENS,	Professeur de Législation Française des Finances et de Science Financière
VIDAL,	Professeur de Droit Privé
DESPAX,	Professeur de Droit Privé
SICARD,	Professeur d'Histoire du Droit
SIORAT,	Professeur de Droit Public
GILLES,	Professeur de Droit Romain
BARRERE J.	Professeur de Droit Privé
MAZERES,	Professeur de Droit Public
DEVILLEBICHOT,	Professeur de Sciences Economiques
SEMPE,	Professeur de Statistique et Méthodes d'Observation
ROUJOU de BOUBEE,	Professeur de Législation Comparée

.../...

MM. RAYBAUD,	Professeur d'Histoire des Faits Economiques et Sociaux
MOURGEON,	Professeur de Droit Public
DELVOLVE,	Professeur de Droit Public
GOUR,	Professeur de Droit Public
DAGOT,	Professeur de Droit Privé
ISAAC,	Professeur de Droit Public Vice-Président de l'Université
SALETTE,	Professeur de Sciences Economiques

PROFESSEUR ASSOCIE

M. CROS,	Professeur de Sciences Economiques

PROFESSEUR SANS CHAIRE

M. POUMAREDE,	Professeur d'Histoire des Institutions
M. SPITERI,	Professeur de Droit Privé
Melle BRUGUIERE,	Professeur d'Histoire des Institutions

MAITRES de CONFERENCES

MM. MOLINIER,	Maître de Conférences Agrégé de Droit Public
BAZERQUE,	Maître de Conférences d'Informatique
PINHAS,	Maître de Conférences de Mathématiques
ARCHER,	Maître de Conférences Agrégé de Sciences Economiques
CABANIS,	Maître de Conférences Agrégé d'Histoire des Institutions
PISTRE,	Maître de Conférences deGGestion
SERLOOTEN,	Maître de Conférences Agrégé de Droit Privé
CAPIAN,	Maître de Conférences Agrégé de Sciences Economiques
BOUYSSOU,	Maître de Conférences Agrégé de Droit Public

MAITRE de CONFERENCES ASSOCIE

M. STOKA,	Maître de Conférences de Mathématiques

CHARGES de COURS

MM. DUPEYRON,	Chargé de Cours de Droit Privé
AMADIO,	Chargé de Cours de Droit Public
CAMPAN,	Chargé de Cours de Sciences Economiques
MOREAUX,	Chargé de Cours de Sciences Economiques
Mme ALCOUFFE,	Chargé de Cours de Sciences Economiques
MM. VILLEVIEILLE,	Chargé de Cours de Droit Public
TOMASIN,	Chargé de Cours de Droit Privé
ASSARAF,	Chargé de Cours de Sciences Economiques
LE POTTIER,	Chargé de Cours de Sciences Economiques
LAVIALLE,	Chargé de Cours de Droit Public
Mme CALMETTE,	Chargé de Cours de Sciences Economiques

.../...

MAITRES-ASSISTANTS

Mme SICARD,	Maître-Assistant d'Histoire des Institutions
M. LABAUVIE,	Maître-Assistant de Sciences Economiques
M. LUDWIG,	Maître-Assistant de Droit Public
Mme CAMBOULIVES,	Maître-Assistant de Droit Privé
Mme HEUZE,	Maître-Assistant de Mathématiques
MM. ARLANDIS,	Maître-Assistant de Gestion
HERSANT,	Maître-Assistant de Sciences Economiques
COULET,	Maître-Assistant de Droit Public
MARICHY,	Maître-Assistant de Droit Public Vice-Président de l'Université
TOURNIE,	Maître-Assistant de Droit Public
LOUBET del BAYLE,	Maître-Assistant de Science Politique
AUBERT,	Maître-Assistant de Gestion
BAUX,	Maître-Assistant de Gestion
LUGAN,	Maître-Assistant de Sociologie
Mme BRUGNES,	Maître-Assistant de Sciences Economiques
MM. ARAGON,	Maître-Assistant de Mathématiques
ALBOUY,	Maître-Assistant de Science Politique
DESMOUTIER,	Maître-Assistant de Gestion
GALAN,	Maître-Assistant de Gestion
REDONNET,	Maître-Assistant d'Anglais
Mle ROUJOU de BOUBEE,	Maître-Assistant de Droit Privé
Mle GUERRIERO,	Maître-Assistant de Droit Privé
M. BIANCONI,	Maître-Assistant de Science Politique

MAITRE-ASSISTANT ASSOCIE

M. KUCERA,	Maître-Assistant de Droit Public

PERSONNEL DETACHE

MM. SIROL,	Professeur (Sciences Economiques) détaché à l'Ambassade de France à Mexico
MOLINS-YSAL,	Professeur (Sciences Economiques) détaché à Montréal
MESTRE,	Maître de Conférences Agrégé (Droit Public) détaché à Tunis

---

Mme ROULLAND,	Conseiller Administratif des Services Universitaires Secrétaire Général de l'Université

PRESIDENT de la THESE :

Suffragants :

L'Université n'entend pas approuver, ni désapprouver les opinions particulières du candidat.

RESUME

La recherche de la solution optimale d'un problème combinatoire est souvent très longue puisque, d'une manière générale, le temps de résolution est une fonction exponentielle de la taille de ce problème.

Dès l'instant où on se place dans un contexte temps réel, c'est-à-dire chaque fois qu'on impose une limite au temps de calcul alloué, les algorithmes d'optimisation classique s'avèrent, à quelques rares exceptions près, impuissants, et on doit se contenter d'utiliser des méthodes heuristiques.

La présente étude porte sur l'évaluation de celles qui sont susceptibles d'apporter une réponse satisfaisante à certains problèmes d'ordonnancement. Trois types d'approches sont étudiés :

- les heuristiques non convergentes,
- les heuristiques convergentes,
- les méthodes d'apprentissage,

L'évaluation de leurs performances est réalisée sur le double critère de la qualité de la solution obtenue et du temps de calcul nécessaire à leur mise en oeuvre.

o
o o
o

Je tiens à remercier :

Monsieur Georges BAZERQUE, Professeur à l'Université des Sciences Sociales de Toulouse, dont l'aide m'a été précieuse tout au long de la réalisation de cette thèse, et qui m'a fait l'honneur d'en présider le Jury.

Monsieur Robert MAHL, qui est à l'origine de ce travail, et qui en a guidé le démarrage alors qu'il était Directeur du département informatique de l'Ecole des Mines de Saint-Etienne.

Messieurs Guy DEVILLEBICHOT et Michel MOREAUX, Professeurs à l'Université des Sciences Sociales de Toulouse, qui ont bien voulu faire partie du Jury.

Monsieur Serge GUIBOUD-RIBAUD, Directeur du département informatique de l'Ecole, qui m'a permis de faire aboutir ce projet.

Monsieur Yves MULLER, Professeur à l'Ecole, qui m'a initié à la recherche opérationnelle, et qui a su me faire part de sa grande expérience dans ce domaine. Les fréquentes discussions que nous avons eues m'ont aidé à formuler et mûrir de nombreuses idées.

Monsieur MONPETIT, Directeur du SESORI, qui a contribué au développement des recherches en ordonnancement à l'Ecole des Mines.

Mes collègues du département informatique, et tout particulièrement Michel NIVAULT et François PETIT-FRESSANCOURT, pour la part active qu'ils ont prise à cette recherche.

Madame BONNEFOY, ainsi que le personnel du Service de Reproduction, pour le soin apporté à la réalisation de ce document.

SOMMAIRE

INTRODUCTION

INTRODUCTION

La "combinatoire" constitue une branche particulière des mathématiques qu'il convient avant tout de caractériser. S'il est difficile d'en fournir une définition précise, du moins peut-on l'approcher de manière intuitive.

Résoudre un problème consiste à attribuer des valeurs à un certain nombre de variables. Celles-ci sont liées par des contraintes, c'est-à-dire des relations qui les rendent dépendantes les unes des autres, et qui définissent l'ensemble des solutions "réalisables" au sens où elles satisfont toutes ces contraintes. Le nombre d'éléments de cet ensemble peut être plus ou moins grand. La solution est :

- vide s'il est égal à 0,
- unique s'il est égal à 1,
- indéterminée s'il est infini.

Lorsqu'il est supérieur à 1, la solution cherchée est le plus souvent caractérisée par un critère ; il s'agit alors de trouver le sous-ensemble des solutions réalisables qui optimise le critère.

Nous dirons d'un problème qu'il est combinatoire lorsque le nombre de solutions réalisables est très grand par rapport au nombre des variables mises en oeuvre.

Exemples :

1 - Problème 1 : "Trouver deux nombres connaissant leur somme S et et leur différence D".

C'est un problème à deux variables X, Y et à deux contraintes :

$$X + Y = S$$

$$X - Y = D$$

Un couple (X, Y) constitue une solution réalisable si :

$$X = \frac{1}{2} (S + D)$$

$$Y = \frac{1}{2} (S - D)$$

Il est clair qu'il n'y a qu'une solution réalisable qui constitue, bien évidemment, la réponse cherchée.

2 - Problème 2 : "Décoder un message"

Un texte a été camouflé à l'aide d'un code qui fait correspondre de manière biunivoque à chaque lettre de l'alphabet un signe. Pour traduire ce message en clair, il suffit de reconstituer ce code. Le problème comporte donc 26 variables. Les contraintes expriment le fait qu'à deux signes différents doivent correspondre deux lettres différentes ; il y en a donc autant que de combinaisons de 2 objets pris parmi 26 soit 325.

Le nombre de solutions réalisables est considérable. Il est égal au nombre de manières d'affecter l'ensemble des lettres à l'ensemble des codes en respectant les contraintes, soit $26 ! \simeq 4. 10^{26}$.

Quant au critère, il est extrêmement simple : les seules traductions intéressantes sont celles qui ont un sens.

Le problème est donc très combinatoire.

Ces deux exemples mettent en évidence la caractéristique essentielle des problèmes combinatoires : le nombre de leurs solutions réalisables est une fonction exponentielle du nombre de variables.

La combinatoire a de tous temps fasciné les mathématiciens. Les carrés magiques par exemple apparaissent pour la première fois dans un livre chinois intitulé "I ching" datant de 2200 avant Jésus-Christ. En Occident, les premiers à s'y être intéressés sont PASCAL et FERMAT au 17ème siècle, qui sont à l'origine des premiers calculs de dénombrement. Le terme "combinatoire" a été utilisé pour la première fois par le philosophe et mathématicien allemand LEIBNITZ qui, dans son traité "Dissertatio de Arte Combinatoria"(1666), entrevit les développements de cette nouvelle discipline dans de nombreuses sciences. Le mathématicien suisse EULER est le promoteur d'une véritable mathématique combinatoire au 18ème siècle : ses travaux en font le père de la théorie des graphes.

A ce stade des connaissances, il était possible dans de nombreux cas de générer n'importe quel élément de l'ensemble des solutions réalisables. Mais il manquait l'outil permettant de parcourir rapidement cet ensemble afin d'en extraire la solution optimale relativement au critère proposé. Depuis une trentaine d'années, cet outil existe : grâce à l'ordinateur et aux progrès constants de l'informatique, l'homme dispose aujourd'hui de moyens de calcul rapides et fiables.

Cependant, le problème n'en est pas résolu pour autant ! Reprenons l'exemple du problème 2, et supposons que notre machine soit capable de construire 10^7 traductions différentes à la seconde. Le temps nécessaire pour parcourir l'ensemble des solutions réalisables serait alors de :

$$\frac{4.10^{26}}{10^7} = 4.10^{19} \text{ secondes}$$

soit approximativement dix milliards de siècles !

Si donc l'ordinateur est indispensable, il ne saurait suffire à la résolution des problèmes combinatoires. Il ne peut-être qu'un outil au service de méthodes plus efficaces que la simple énumération.

1 - LES PROBLEMES D'ORDONNANCEMENT.

Avant d'aborder la présentation de ces méthodes, il nous faut définir plus précisément le type de problèmes utilisés comme support à cette étude.

L'aspect combinatoire intervient dans de nombreux domaines. Citons par exemple :

- la recherche opérationnelle,
- la linguistique
- la traduction et la documentation automatique,
- la reconnaissance des formes,
- l'intelligence artificielle.

Dans un but d'unification du vocabulaire et des notations, il nous a paru souhaitable de nous limiter à un seul d'entre eux, la recherche opérationnelle, et plus particulièrement l'ordonnancement. Ce choix n'est pas aussi restrictif qu'il le paraît : beaucoup de problèmes sont formalisables en termes d'ordonnancement.

En effet, ordonnancer n variables c'est déterminer un ordre total réalisable, compte tenu d'un certain nombre de contraintes, et qui satisfait au mieux un critère donné. Nous ne nous intéressons qu'aux problèmes non stochastiques, ce qui élimine les théories des queues et des files d'attente. La classe plus restreinte, qui nous préoccupe, recouvre pourtant des problèmes très divers dont les applications pratiques sont très nombreuses. Et en particulier :

1.1 - L'ordonnancement d'un atelier

On dispose d'un certain nombre de machines pour fabriquer des pièces. Chaque pièce doit passer sur les machines de l'atelier dans un ordre déterminé par sa gamme de fabrication. La durée d'exécution de chaque tâche, constituée par le passage d'une pièce sur une machine, est connue à priori. Les contraintes qui interviennent le plus souvent sont les suivantes :
- une machine ne peut traiter qu'une pièce à la fois.
- toute tâche commencée doit aller sans interruption jusqu'à son complet achèvement.

Il s'agit de déterminer pour chaque machine l'ordre de passage des pièces qui lui sont affectées en cherchant à minimiser :
- soit la somme pondérée des retards à la sortie de l'atelier.
- soit le retard maximum.
- soit encore le temps global d'occupation de l'atelier.

On distingue dans la pratique deux types de problèmes :
- ceux où le temps de réemploi est nul ; dans ce cas, aucun délai n'est nécessaire pour adapter une machine à une nouvelle pièce.
- ceux où interviennent un délai de réemploi : le temps total d'occupation de chaque machine M_j de l'atelier est alors égal à la somme des temps d_{ij} de fabrication de chacune des pièces i affectées à cette machine, augmentée des délais de réemploi. Le temps d'adaptation au traitement d'une nouvelle pièce i' dépend en général de i', de la pièce i qui vient d'être achevée, et de la machine sur laquelle a lieu le traitement.

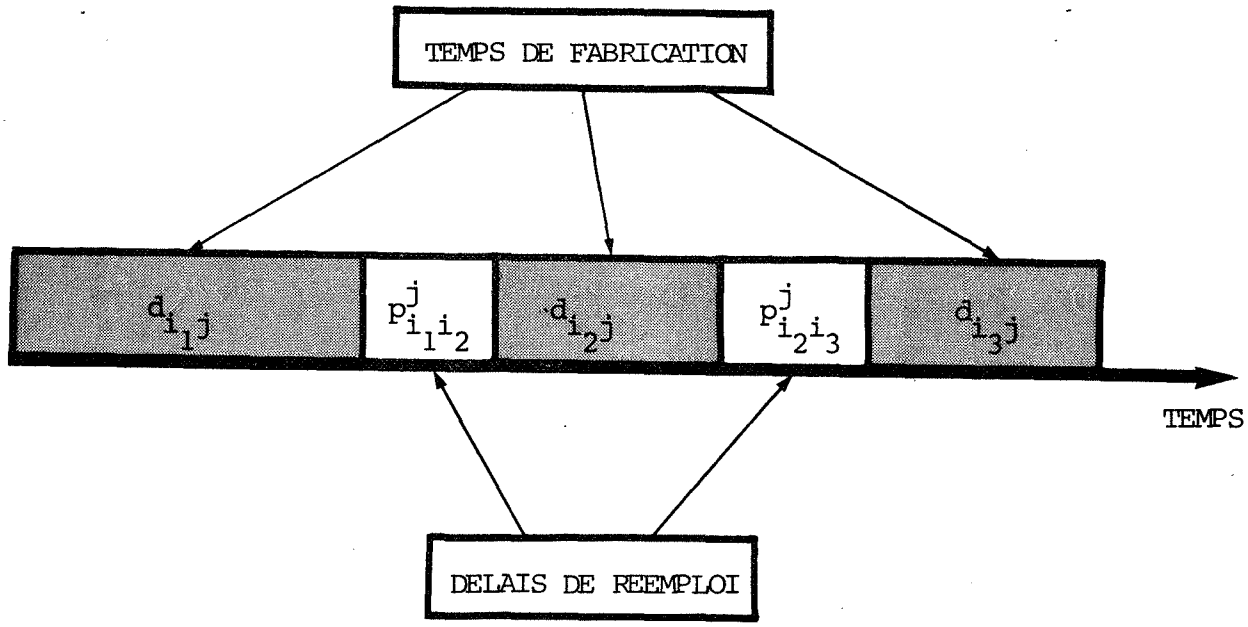


Diagramme de fonctionnement de la machine Mj

1.2. - L'ordonnancement des travaux dans un ordinateur.

Les ordinateurs modernes partagent leur activité entre l'exécution de plusieurs tâches :

- Dans un système à multiprogrammation classique, plusieurs travaux résidant en mémoire centrale se font concurrence pour obtenir les services de l'unité centrale ou des périphériques.

  Il s'agit alors d'optimiser :
  . soit l'utilisation globale des ressources de manière à augmenter le rendement du système.
  . soit l'avancement pondéré de l'ensemble des travaux de manière à diminuer les temps d'attente imposés aux utilisateurs.

- Dans un système temps réel, les travaux sont caractérisés par des dates critiques ("deadline") auxquelles ils doivent être impérativement achevés. Chacun d'entre eux est réactivé au bout d'une certaine période. Il s'agit de savoir s'il faut lui allouer ou non la ressource qu'il demande. L'algorithme le plus couramment utilisé est le

"shortest deadline first" qui affecte la ressource au travail dont la date critique est la plus rapprochée. L'optimalité de cet algorithme a été prouvée récemment par LABETOULLE [8].

- Dans un système temps partagé, le critère d'allocation est plus complexe. Il tient compte à la fois de la longueur probable du travail, de son temps d'activité passée, etc...

- Dans un réseau d'ordinateurs à commutation de paquets, chaque commutateur doit prendre des décisions de manière extrêmement rapide à la fois sur les paquets à envoyer et sur les lignes qu'ils doivent emprunter. Dans ce cas, on tient compte des priorités des paquets, et on cherche à minimiser la saturation du réseau.

Dans tous ces exemples, les décisions concernant l'allocation des ressources et l'ordonnancement doivent être prises en une fraction de seconde, voire quelques millisecondes. Ceci a justifié sur certains ordinateurs (CDC 6600) l'emploi de processeurs spécialisés pour résoudre ces problèmes.

1.3. L'organisation du trafic.

Le problème de la régulation du trafic est appelé à prendre une importance considérable dans le monde moderne. Pour échapper à la congestion des réseaux, il semble que seules les deux solutions suivantes puissent être envisagées. Elles consistent :

- soit à multiplier les lignes de communication, ce qui est coûteux et peu satisfaisant dans la mesure où le rendement de l'ensemble reste faible.
- soit à organiser l'écoulement du trafic sur le réseau existant.

L'ordonnancement trouve ainsi un vaste champ d'application dans le domaine de la régulation du trafic quel qu'il soit : routier, ferroviaire, aérien, téléphonique, etc... Ce type de problème peut être formalisé comme celui de l'ordonnancement d'un atelier, les pièces représentant des mobiles (automobiles, trains, avions, appels,...) et les machines des supports (routes, voies ferrées, couloirs aériens, lignes téléphoniques,...).

2 - LA CONTRAINTE TEMPS PEEL

Dans la pratique, les délais alloués pour la résolution d'un problème peuvent être très différents, selon le problème envisagé.

S'il s'agit d'organiser le programme de fabrication mensuel d'une unité de production, ou encore de calculer les heures de départ des trains réguliers sur une période d'un semestre, on peut en général aborder le problème bien avant que la solution ne soit exploitée sur le terrain. Le temps alloué est pratiquement infini et il est alors possible de trouver la solution optimale à l'aide d'un algorithme d'optimisation classique. Ces méthodes sont connues, elles ne font pas l'objet de la présente étude.

Mais, il existe également des cas où la solution doit être trouvée extrêmement rapidement, par exemple lorsqu'interviennent des perturbations : panne d'une machine, lancement en fabrication de commandes urgentes non planifiées, prise en compte de trains supplémentaires, mise hors service de tronçons de voies ferrées. Il faut alors trouver un nouvel ordonnancement suffisamment rapidement, en "temps réel", pour ne pas introduire des retards artificiels dans le processus de fabrication. Concrètement, cela signifie que les temps de calcul devront être de l'ordre de quelques minutes au maximum.

Or les problèmes d'ordonnancement sont parmi les plus combinatoires, et compte tenu de cette contrainte temps réel, il est bien évident qu'il ne sera pas possible d'explorer l'ensemble des solutions réalisables pour y détecter l'optimale.

3 - LES APPROCHES POSSIBLES.

Dans ces conditions, les méthodes à envisager pour résoudre un tel problème doivent se limiter à l'exploration d'un sous-ensemble très réduit de ces solutions.

Il n'existe pas de théorie générale qui englobe tous les problèmes d'ordonnancement et qui soit susceptible d'apporter une réponse satisfaisante à chacun d'eux. Les études les plus récentes [13] font état d'un catalogue

d'algorithmes plus ou moins adaptés à chaque type de problèmes.La plupart d'entre eux sont inefficaces dans un contexte temps réel car leur mise en oeuvre serait beaucoup trop longue. A la suite de cette constatation, les recherches se sont orientées dans deux voies divergentes :

1) Approches faisant intervenir un matériel spécifique : l'une des solutions possibles pour accélérer le déroulement d'un algorithme consiste à le câbler en partie ou en totalité sur une unité spécialisée. Ceci suppose, bien entendu, qu'il soit suffisamment général pour pouvoir s'appliquer à une classe étendue de problèmes. Les études menées en France dans ce domaine ont conduit, dans les années 1965-1970, à la réalisation de prototypes de machines dont les performances sont évaluées dans la thèse de M. NIVAULT [12].

2) Approches purement logicielles : Une autre solution envisageable consiste à n'employer que les méthodes les plus rapides ou à adapter celles qui peuvent l'être à la contrainte temps réel. Leur étude fait l'objet de la présente recherche. Notre propos n'est pas de les évaluer toutes, mais plutôt de les regrouper en classes dont nous tenterons de définir les caractéristiques générales. La première distinction que l'on peut faire porte sur la qualité de la solution obtenue. Nous distinguerons :

Les méthodes globales.

Ce sont celles qui permettent à coup sûr d'atteindre l'optimum. Leur principe consiste à ne construire qu'une solution réalisable qui est en même temps la solution optimale. Après avoir formalisé l'ensemble des problèmes d'ordonnancement, nous montrerons dans le chapitre 1 que quelques uns d'entre eux peuvent être résolus globalement en temps réel.

Les méthodes heuristiques.

Le plus souvent, on doit se contenter d'utiliser des heuristiques qui produisent de bonnes solutions, sans en garantir l'optimalité. Nous les classerons en deux grandes catégories :

- les heuristiques non convergentes : une seule solution réalisable est construite, dont on ne peut rien dire de la qualité. Le chapitre 2 est consacré à la présentation de celles qui sont utilisées pour résoudre certains problèmes d'ordonnancement.

- les heuristiques convergentes : contrairement aux précédentes, ces méthodes explorent un sous-ensemble relativement vaste de solutions réalisables. Leur étude est faite au chapitre 3.

Nous aborderons ensuite la classe d'heuristiques un peu particulières que sont les heuristiques d'apprentissage. Il s'agit de méthodes qui généralisent les précédentes et qui permettent le plus souvent d'en affiner les résultats. Leur principe consiste à itérer un certain nombre de fois une méthode de résolution, convergente ou non, et à extraire des résultats obtenus des caractéristiques qui sont utilisées pour construire la solution du problème. Leur présentation fait l'objet du chapitre 4.

CHAPITRE 1

LES PROBLEMES D'ORDONNANCEMENT

METHODES GLOBALES EN TEMPS REEL

CHAPITRE I

LES PROBLEMES D'ORDONNANCEMENT METHODES GLOBALES EN TEMPS REEL

Pour clarifier l'exposé, nous ne parlerons désormais que d'ordonnancement d'atelier. Cette convention a le double mérite d'englober tous les types de problèmes connus, à condition d'adopter une formalisation suffisamment large, et de permettre de définir une notation précise.

1 - FORMALISATION

Nous pouvons dès lors poser le problème de la manière suivante : "Etant données n pièces qui doivent passer sur m machines dans un ordre donné et sous certaines contraintes, quel est, relativement à un critère donné, l'ordre optimal dans lequel chaque machine traite l'ensemble des pièces".

Les variables qui définissent la solution sont du type x_i^j, rang de passage de la pièce i sur la machine j. Il faut donc pour chacune de ces machines, affecter aux variables $\{x_1^j, x_2^j, \ldots, x_n^j\}$ des valeurs obtenues par permutation des entiers $\{1, 2, \ldots, n\}$ de sorte que la solution ainsi obtenue soit réalisable et qu'elle optimise le critère.

Au premier abord, un problème est caractérisé par un ensemble de données. Pour qu'il soit complètement déterminé, il faut encore en préciser les contraintes et le critère.

1.1. LES DONNEES

On suppose que sont connus :

- le nombre n des pièces $P_1, P_2, \ldots, P_n$.
- le nombre m des machines $M_1, M_2, \ldots, M_m$.

- les durées d'exécution des tâches. Une tâche est déterminée de façon unique par le passage d'une pièce P_k sur une machine M_l. Elle est notée (P_k, M_l) ou plus simplement (k,l). La durée correspondante est d_{kl}.

- les gammes de fabrication de chacune des pièces. $G(i,j) = k$ signifie que la j^e machine de la gamme de P_i est M_k. On en distingue essentiellement trois types :

  . G1 : toutes les gammes sont identiques ("flow-shop"). Cela ne veut pas dire que les machines traitent nécessairement les pièces dans le même ordre.

  . G2 : toutes les gammes sont identiques et en outre les machines traitent les pièces dans le même ordre.

  . G3 : les gammes sont différentes ("job-shop").

- éventuellement, les durées de réemploi p^i_{jk}, temps d'adaptation nécessaire au passage de la fabrication de la pièce j à celle de la pièce k sur la machine i.

1.2. LES CONTRAINTES.

Les plus fréquemment rencontrées sont les suivantes :

(P1) L'ensemble des pièces est connu et fixé.

(M1) L'ensemble des machines est connu et fixé.

(P2) Toutes les pièces sont disponibles au même instant (= 0).

(M2) Toutes les machines sont disponibles au même instant (= 0).

(P3) Toutes les pièces restent disponibles pendant un temps infini (pas de dates critiques).

(M3) Toutes les machines restent disponibles pendant un temps infini (pas de grèves ni de pannes).

(P4) Chaque pièce est dans l'un des trois états suivants : en attente de sa prochaine machine, en traitement sur une machine, ou complètement terminée.

(M4) Chaque machine est dans l'un des trois états suivants : en attente d'une pièce, travaillant sur une pièce, ou ayant terminé sa dernière pièce.

(P5) Toutes les pièces sont différentes.

(M5) Toutes les machines sont différentes.

(P6) Toutes les pièces sont également importantes (pas de pièces prioritaires).

(M6) Toutes les machines sont également importantes.

(P7) Chaque pièce passe par toutes les machines de sa gamme.

(M7) Chaque machine traite toutes les pièces qui lui sont assignées.

(P8) Chaque pièce occupe une seule machine à la fois.

(M8) Chaque machine traite une pièce à la fois.

(PM1) Toutes les durées d'exécution sont connues et fixées (c'est-à-dire indépendantes de l'ordre dans lequel les pièces passent sur les machines ; autrement dit, pas de durée de réemploi).

(PM2) Toute tâche commencée doit aller jusqu'à son complet achèvement (non préemptibilité).

(PM3) Les gammes de fabrication sont connues et fixées.

Certaines de ces contraintes sont beaucoup plus importantes que d'autres.(P2) et (M2) ne modifient pas de manière significative la nature du problème posé et donc les méthodes susceptibles de le résoudre. Par contre (P1) et (M1) permettent de faire la distinction entre les problèmes statiques (déterministes) et les problèmes dynamiques (stochastiques).

1.3. LES CRITERES.

On peut les classer en quatre groupes principaux :

- critères basés sur les dates d'achèvement et les temps de passage.
- critères basés sur les retards.
- critères basés sur les en-cours et le coefficient d'utilisation.
- critères basés sur les durées de réemploi.

1.3.1. - Critères basés sur les dates d'achèvement et les temps de passages.

Il nous faut d'abord définir un certain nombre de grandeurs ; on suppose que l'on a n pièces $(P_1, \ldots P_n)$ et m machines $(M_1, \ldots, M_m)$. La durée d'exécution de la tâche (P_k, M_l) est notée d_{kl}.

C_k = date à laquelle la pièce P_k est disponible (date au plus tôt à partir de laquelle elle peut être mise en fabrication). Sous la contrainte (P2), $C_k = 0 \ \forall k$.

a_{kl} = temps d'attente de la pièce P_k devant la machine M_l.

$A_k = \sum_{l=1}^{m} a_{kl}$ temps total d'attente de P_k.

$d_k = \sum_{l=1}^{m} d_{kl}$ durée de fabrication de P_k.

T_k = date d'achèvement de P_k.

F_k = temps total que P_k passe dans l'atelier.

Il existe des relations élémentaires entre ces grandeurs :

$$T_k = C_k + A_k + d_k \qquad (1)$$

$$F_k = A_k + d_k \qquad (2)$$

Nous pouvons maintenant définir les critères les plus fréquemment utilisés :

(C1) Minimiser le maximum des dates d'achèvement :

$$T_{max} = \max_k \{T_k\}$$

(C2) Minimiser le maximum des temps de passage, ou encore le temps d'activité de l'atelier :

$$F_{max} = \max_k \{F_k\}$$

(C3) Minimiser le maximum des temps d'attente :

$$A_{max} = \max_k \{A_k\}$$

Trois nouveaux critères portent sur les moyennes, ou plus généralement les moyennes pondérées. Le coefficient α_k affecté à la pièce P_k mesure son importance. Sous la contrainte (P6), $\alpha_k = 1 \ \forall k$.

(C4) Minimiser la moyenne pondérée des dates d'achèvement :

$$\bar{T} = \sum_k \alpha_k T_k$$

(C5) Minimiser la moyenne pondérée des temps de passage:

$$\bar{F} = \sum_k \alpha_k \; F_k$$

(C6) Minimiser la moyenne pondérée des temps d'attente :

$$\bar{A} = \sum_k \alpha_k \; F_k$$

Les égalités (1) et (2) montrent que les critères (C4), (C5) et (C6) sont équivalents. Les critères (C1) et (C2) le sont également si $C_k = o \; \forall k$, ce que nous supposerons désormais sauf en cas de convention explicite contraire.

1.3.2. - Critères basés sur les retards.

La contrainte (P3) n'est plus à retenir. A chaque pièce P_k est affectée une date critique f_k. On pose :

$Q_k = T_k - f_k$ retard algébrique de P_k.

$R_k = \max (O, Q_k)$ retard de P_k.

$S_k = \max (O, -Q_k)$ avance de P_k.

Les critères sont alors les suivants :

(C7) Minimiser le maximum du retard algébrique :

$$Q_{max} = \max_k \{Q_k\}$$

(C8) Minimiser le maximum du retard :

$$R_{max} = \max_k \{R_k\}$$

(C9) Maximiser le maximum de l'avance :

$$S_{max} = \max_k \{S_k\}$$

(C10) Minimiser le retard algébrique moyen :

$$\bar{Q} = \sum_k \alpha_k \; Q_k$$

(C11) Minimiser le retard moyen :

$$\bar{R} = \sum_k \alpha_k \quad R_k$$

(C12) Maximiser l'avance moyenne :

$$\bar{S} = \sum_k \alpha_k \quad S_k$$

La relation de définition de Q_k montre que (C10) est équivalent à (C4), (C5), (C6).

1.3.3. Critères basés sur les en-cours et le coefficient d'utilisation.

Il est possible de juger la qualité d'un ordonnancement en examinant ce qui se passe tout au long du processus de production. On introduit :

Np(t) = nombre de pièces achevées à l'instant t
Na(t) = nombre de pièces en attente à l'instant t
Nf(t) = nombre de pièces en cours de fabrication à l'instant t.
Wp(t) = travail fait, c'est-à-dire somme des temps d'exécution des tâches terminées à l'instant t.
Wa(t) = travail à faire : somme des temps d'exécution des tâches qu'il reste à effectuer à l'instant t.
Wf(t) = travail en cours : somme des temps des tâches en cours d'exécution à l'instant t.

Par définition :

$$Np(t) + Na(t) + Nf(t) = n$$
$$Wp(t) + Wa(t) + Wf(t) = \sum_{k,l} d_{kl}$$

Si on considère les moyennes de ces grandeurs sur l'intervalle (O, F_{max}), on voit que :

$\overline{Na} + \overline{Nf}$ est le niveau moyen des en-cours.
$\bar{N}p$ est le niveau moyen des produits finis.

Les critères sont les suivants :

(C13) Maximiser $\overline{Np}$

(C14) Minimiser $\overline{Na} + \overline{Nf}$

(C15) Minimiser $\overline{Na}$

(C16) Maximiser $\overline{Nf}$

(C17) Maximiser $\overline{Wf}$

(C18) Minimiser la somme des temps morts :

$$mF_{max} - \sum_{k,l} d_{kl}$$

(C19) Maximiser le coefficient d'utilisation

$$U = \sum_{k,l} d_{kl} / m F_{max}$$

Il existe des relations entre ces critères. En effet :

$$\overline{Na} + \overline{Nf} = (F1 + F2 + \ldots + Fn) / F_{max} = n\bar{F}/F_{max}$$

d'où

$$\bar{Np} = n - (\bar{Na} + \bar{Nf}) = n(1 - \bar{F}/F_{max})$$

Ces deux égalités montrent l'équivalence des critères (C13) et (C14). On montre également que :

$$\overline{Nf} = \sum_{k,l} d_{kl} / F_{max}$$

$$\overline{Wf} = \sum_{k,l} d_{kl}^{2} / F_{max}$$

On en conclut l'identité des critères (C2), (C16), (C17), (C18), (C19)

1.3.4. - Critères sur les durées de réemploi.

On considère un problème à n pièces et une machine où, à la durée d'exécution d'une tâche, il faut ajouter le temps d'adaptation ou de réemploi de la machine lorsqu'elle passe d'une pièce à une autre. Le temps d'occupation de l'atelier dépend alors de l'ordre de passage des pièces. Le seul critère utilisé est la minimisation de la somme des durées de réemploi :

(C20) Minimiser la somme des durées de réemploi :

$$P = \sum_{i,j} p_{ij}$$

1.3.5. - Récapitulation.

Nous avons vu qu'un certain nombre de critères étaient équivalents entre eux. Il est donc possible de les regrouper en classes que nous caractériserons par le critère le plus simple à formuler.

CLASSE	CRITERE CARACTERISTIQUE
(C1), (C2), (C16), (C17), (C18), (C19)	Minimiser F_{max}, maximum des temps de passage.
(C3)	Minimiser A_{max}, maximum des temps d'attente.
(C4), (C5), (C6), (C10)	Minimiser $\bar{F}$, moyenne pondérée des temps de passage.
(C7)	Minimiser Q_{max}, maximum du retard algébrique.
(C8)	Minimiser R_{max}, maximum du retard.
(C9)	Maximiser S_{max}, maximum de l'avance.
(C11)	Minimiser $\bar{R}$, moyenne pondérée du retard.
(C12)	Maximiser $\bar{S}$, moyenne pondérée de l'avance.
(C13), (C14)	Maximiser $\bar{Np}$, niveau moyen des produits finis.
(C15)	Minimiser $\bar{Na}$, nombre moyen des pièces en attente.
(C20)	Minimiser P, somme des durées de réemploi.

1.4. CARACTERISATION DES PROBLEMES.

Un problème est parfaitement défini lorsque sont précisés les données, les contraintes qui entrent en jeu, et le critère à optimiser. C'est pourquoi, CONWAY, MAXWELL et MILLER [2] ont proposé de le caractériser par une notation compacte du type :

$$n \mid m \mid G \mid D1, \ldots, Dn \mid C$$

dans laquelle :

- n et m désignent respectivement le nombre de pièces et de machines ;
- G caractérise les gammes ;
- D1,..., Dn définissent les contraintes. Dans la plupart des cas, les contraintes à prendre en compte représentent la presque totalité de celles que nous avons énumérées. Il a donc paru plus simple de ne citer dans la liste des Di que celles qui n'interviennent pas ;
- C désigne le critère utilisé.

Il existe un nombre considérable de classes de problèmes, et il n'est pas possible, dans le cadre de cette étude, de les examiner toutes. Nous n'en avons retenu que deux, très classiques, afin d'évaluer les performances des méthodes de résolution heuristique.

- n|1|PM1|C20 : n pièces doivent passer sur une machine avec des durées de réemploi dont il faut minimiser la somme. Ce problème est connu en recherche opérationnelle sous le nom de "voyageur de commerce". Il est en effet isomorphe à celui d'un voyageur qui se propose de visiter n villes en passant une fois et une seule par chacune d'elles de manière à minimiser la somme des distances parcourues. Trois cas numériques sont donnés en Annexe 1 : les valeurs de n sont respectivement 33, 42, et 57. Pour chacun d'eux est indiquée la solution optimale.

- n|m|G3|C1 : C'est un ordonnancement d'atelier à n pièces et m machines ; les gammes sont toutes différentes et le critère à optimiser est le temps d'activité de l'atelier. Trois cas ont été traités qui sont cités en annexe 2, et pour lesquels les valeurs de n|m sont 6|6, 10|10, 20|5. Ce problème, connu sous le nom de "job-shop", sera désormais appelé ordonnancement d'atelier.

Ces deux problèmes sont très combinatoires. Les nombres de leurs solutions réalisables sont respectivement donnés par les formules suivantes :

- voyageur de commerce : n ! solutions. Ce nombre est de l'ordre de $(\frac{n}{e})^n$ (1), où e désigne la base des logarithmes népériens.
- Ordonnancement d'atelier : il s'agit en fait de résoudre m problèmes de voyageur imbriqués. Il y a donc en principe $(n!)^m$ solutions réalisables soit, approximativement $(\frac{n}{e})^{nm}$.

---

(1) La valeur de n! peut être approchée par la formule de Stirling :

$$n! \simeq n^n e^{-n} \sqrt{2 \pi n} \left(1 + \frac{1}{12n}\right)$$

Il est intéressant de remarquer que ces nombres varient exponentiellement avec le nombre des variables de chaque problème. Si nous les évaluons numériquement dans deux des cas concrets que nous nous proposons de résoudre, nous obtenons :

PROBLEME	n	m	NOMBRE DE SOLUTIONS REALISABLES
voyageur	57	1	$4 .10^{76}$
atelier	20	5	$8 .10^{91}$

Compte tenu de la contrainte temps réel qui nous est imposée, il est bien évident qu'il ne sera pas possible d'explorer l'ensemble de ces solutions pour y détecter l'optimale. En effet, le temps nécessaire pour effectuer cette exploration serait en première approximation proportionnel au nombre de solutions réalisables, et donc incompatible avec cette contrainte dès lors que le nombre de variables est suffisamment grand.

2 - LES METHODES GLOBALES.

Leur principe commun consiste à ne construire qu'une solution réalisable qui est en même temps la solution optimale. On peut donc s'attendre à ce que le temps de calcul nécessaire à leur mise en oeuvre soit très court. Malheureusement, très peu de problèmes peuvent être résolus par de telles méthodes. A notre connaissance, il n'en existe que trois, qui sont tous du type n|1 ou n|2.

2.1. METHODE DE SMITH [15]

La classe de problèmes traitée est :

$$n|1|P6|C4$$

où le critère à minimiser est la moyenne pondérée des dates d'achèvement. SMITH a montré que la solution optimale est donnée par la séquence $\{i_1, i_2, \ldots, i_n\}$, où i_j est le rang de passage de la pièce j, telle que :

$$d_{i_1} / \alpha_{i_1} \leq d_{i_2} / \alpha_{i_2} \leq \ldots \leq d_{i_n} / \alpha_{i_n}$$

Preuve

Supposons la séquence $\{i_1, i_2, \ldots, i_n\}$ construite d'après la règle précédente et construisons une nouvelle séquence en permutant le rang des pièces i_k et i_{k+1}. Les seules dates d'achèvement affectées par cette transformation sont celles de ces deux pièces. La nouvelle séquence ne peut être meilleure que la première que si :

$$\alpha_{i_k} \sum_{j=1}^{k} d_{i_j} + \alpha_{i_{k+1}} \sum_{j=1}^{k+1} d_{i_j} > \alpha_{i_{k+1}} \left(\sum_{j=1}^{k-1} d_{i_j} + d_{i_{k+1}} \right) + \alpha_{i_k} \sum_{j=1}^{k+1} d_{i_j}$$

Par simplification cette expression devient :

$$\alpha_{i_{k+1}} \; d_{i_k} > \alpha_{i_k} \; d_{i_{k+1}}$$

ou encore :

$$d_{i_k} / \alpha_{i_k} > d_{i_{k+1}} / \alpha_{i_{k+1}}$$

ce qui est contraire à notre hypothèse.

La séquence de SMITH est donc bien optimale.

Temps de calcul.

Il correspond au temps qui est nécessaire pour effectuer le tri de n nombres ; il est donc proportionnel à n Log n.

2.2. METHODE DE JACKSON [4]

Elle permet de résoudre les problèmes :

$$n|1|P3|C7 \text{ et } n|1|P3|C8$$

où il s'agit de minimiser le maximum du retard algébrique ou absolu. La séquence optimale est définie par la suite d'inégalités :

$$f_{i_1} \leq f_{i_2} \leq \ldots \leq f_{i_n}$$

f_{i_j} étant la date critique de la pièce i_j.

Preuve

Supposons que le critère soit C7 et construisons une séquence en permutant le rang des pièces i_k et i_{k+1} dans la séquence de JACKSON. Le nouveau retard de la pièce i_k s'écrit :

$$Q_k^{(2)} = \sum_{j=1}^{k+1} d_{i_j} - f_{i_k}$$

Les retards des pièces i_k et i_{k+1} dans la séquence initiale s'écrivaient respectivement :

$$Q_k^{(1)} = \sum_{j=1}^{k} d_{i_j} - f_{i_k}$$

$$Q_{k+1}^{(1)} = \sum_{j=1}^{k+1} d_{i_j} - f_{i_{k+1}}$$

d'où on déduit :

$$Q_k^{(2)} > Q_k^{(1)}$$

$$Q_k^{(2)} > Q_{k+1}^{(1)} \longleftrightarrow f_{i_k} < f_{i_{k+1}}$$ ce que nous avons supposé.

Il vient donc :

$$Q_k^{(2)} > \max\ (Q_k^{(1)}, Q_{k+1}^{(1)})$$

D'où on peut conclure que le retard maximum de la première séquence est nécessairement inférieur ou égal à celui de la seconde. La méthode apporte également la solution optimale lorsque le critère à minimiser est $R_{max} = \max\ (O, Q_{max})$ (C8)

Temps de calcul

Il est en n Log n.

2.3. METHODE DE JOHNSON [6].

Le problème étudié est du type :

$$n|2|G1|C1$$

Les gammes sont identiques et le critère à minimiser est le maximum des temps de passage.

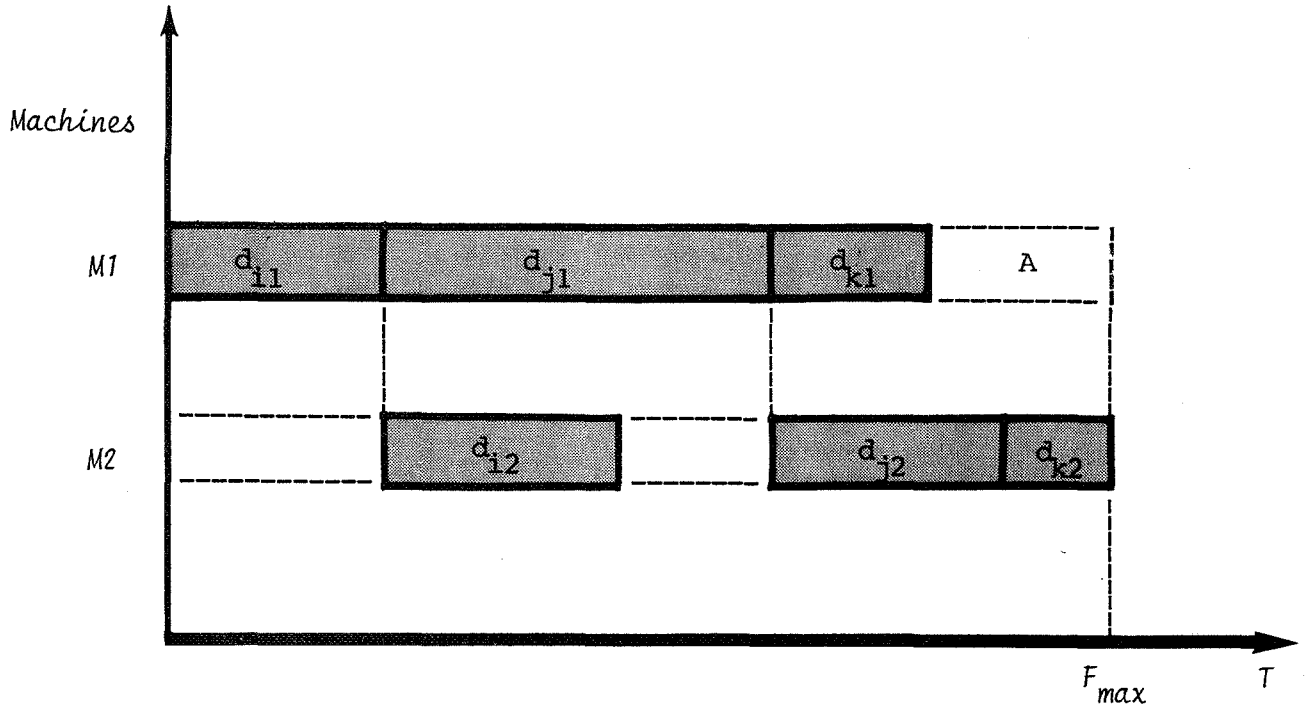
JOHNSON a montré que la séquence optimale est construite d'après la règle :

> La pièce i précède la pièce j si :
> $\min (d_{i1}, d_{j2}) < \min (d_{j1}, d_{i2})$

Preuve

Nous ne donnerons qu'une preuve intuitive de l'algorithme proposé. Il est clair que dans la séquence optimale, la machine 1 est utilisée de manière continue et que les ordres de passage des pièces sur les deux machines sont identiques.

Représentons graphiquement une solution qui satisfait à ces deux conditions :



On voit sur ce diagramme de GANTT que pour diminuer F_{max} on peut agir en réduisant les temps morts de la machine 2 et le temps d'inoccupation A de la machine 1. On a donc tout intérêt à :

- faire passer en tête les pièces dont les temps d'exécution sur la première machine sont les plus courts (réduction des temps morts).
- traiter en queue celles dont les temps de fabrication sur la seconde sont les plus courts (réduction de A).

Temps de calcul.

Il est en n Log 2n

JACKSON [5] a généralisé la méthode précédente au cas où les pièces passent sur les machines dans un ordre quelconque (problème n|2|G3|C1). Il suffit de diviser l'ensemble des pièces en quatre sous-ensembles :

- {1} : sous-ensemble des pièces qui ne passent que sur la machine 1.
- {2} : " " " " " " " " " " " " 2.
- {1,2} : sous-ensemble des pièces qui passent sur la machine 1 puis sur la 2.
- {2,1} : " " " " " " " " " 2 " " " 1.

Dans chacun des sous-ensembles {1,2} et {2,1} les pièces sont ordonnées en appliquant la méthode de JOHNSON et sans tenir compte des autres pièces. L'ordonnancement optimal est alors donné par :

machine 1 : {1,2} , {1} , {2,1}
machine 2 : {2,1} , {2} , {1,2}

2.4. INTERET DE CES METHODES

L'efficacité des méthodes globales que nous avons présentées est évidente : il est difficile d'imaginer des algorithmes plus rapides, et la solution produite est optimale. Cependant on ne peut considérer qu'elles apportent une réponse satisfaisante au problème d'ordonnancement en temps réel puisque leurs cas d'application sont très spécifiques et limités, et que le plus souvent, il faut avoir recours aux méthodes heuristiques.

CHAPITRE 2

LES HEURISTIQUES NON CONVERGENTES

C H A P I T R E II

LES HEURISTIQUES NON CONVERGENTES

Comme nous l'avons dit, le principe général de ces méthodes consiste à ne construire qu'une seule solution réalisable. C'est la raison pour laquelle elles sont qualifiées de non convergentes : la solution proposée est unique et n'est donc pas susceptible d'améliorations. Les algorithmes mis en oeuvre sont extrèmement simples; le plus souvent on se contente d'effectuer un tri sur une certaine grandeur. Ils sont conçus pour résoudre un type de problème bien précis dont ils utilisent toutes les spécificités pour en fournir une solution qui soit la meilleure possible et dans le délai le plus bref. A chacun de ces types correspond donc un certain nombre d'heuristiques plus ou moins adaptées. Celles-ci sont qualifiées d'itérables ou de non itérables suivant qu'on peut les utiliser plusieurs fois ou non pour résoudre un même problème.

1 - LES HEURISTIQUES NON ITERABLES.

1.1. PRINCIPE

La solution produite est indépendante de toute initialisation et n'est fonction que des données. Les algorithmes varient d'un problème à l'autre. On en connaît un certain nombre pour résoudre l'ordonnancement d'atelier.

1.2. APPLICATION AUX PROBLEMES D'ORDONNANCEMENT.

Supposons donc que nous ayons à résoudre un problème d'atelier à n pièces et m machines dans lequel chaque pièce doit passer une fois et une seule sur chacune des m machines dans un ordre déterminé par sa gamme

de fabrication. Le critère à minimiser est le temps d'activité de l'atelier. Compte tenu des contraintes que nous avons admises, et en particulier de la non préemptibilité, la résolution d'un tel problème passe par la prise de n x m décisions successives : à chaque fois qu'une machine se libère, il faut choisir une pièce parmi celles qui sont en attente de cette machine à l'instant considéré (nous dirons qu'elles sont en en-cours d'entrée), de manière à lui faire poursuivre son travail.

Toute la difficulté consiste donc à effectuer ce choix, dont dépend bien évidemment la qualité de la solution finale. Chaque pièce dans l'en-cours d'entrée peut être caractérisée par une grandeur physique telle que :

- la durée de son passage sur la machine,
- son temps de traitement résiduel avant sa complète fabrication,
- etc...

Les heuristiques proposées ne diffèrent que par la grandeur caractéristique choisie et par le procédé de sélection sur cette grandeur. Après avoir décrit leur principe de fonctionnement, nous examinerons les résultats numériques qu'elles ont fournis.

- Heuristique CD

CD = plus Courte Durée

Les pièces en en-cours sont caractérisées par la durée de la tâche correspondante (temps de passage de la pièce sur la machine). On prend la décision de faire passer celle dont la durée est minimale.

Cet algorithme est particulièrement approprié chaque fois que les gammes présentent un "goulot d'étranglement" en début de fabrication : si parmi les m machines, un petit nombre d'entre elles se trouvent systématiquement en tête de gamme, on a tout intérêt à sélectionner les pièces dont les temps de passage sont les plus courts afin d'alimenter au plus vite les autres machines de l'atelier.

- Heuristique LD.

LD = plus Longue Durée

La tâche choisie est celle dont la durée est la plus longue. La méthode fournit de bons résultats chaque fois que le nombre de pièces est

petit devant le nombre de machines. Dans ce cas, en effet, il n'y a pratiquement pas de files d'attente dans l'en-cours d'entrée, et le temps minimal d'activité de l'atelier est en général peu supérieur au temps de fabrication le plus long. Il est donc naturel de favoriser les tâches correspondantes.

- Heuristique CTR.

CTR = plus Court Temps de fabrication Résiduel.

La grandeur sur laquelle est opéré le choix n'est plus la durée de la tâche, mais le temps de fabrication résiduel ; à chaque instant, il est égal au temps minimal au bout duquel on peut espérer que la pièce soit terminée ; c'est donc la somme des temps de fabrication de la pièce sur les machines de sa gamme sur lesquelles elle doit encore passer jusqu'à sa complète réalisation. L'heuristique CTR sélectionne la pièce pour laquelle ce temps est minimal. Les cas d'application sont en pratique les mêmes que CD.

- Heuristique LTR.

LTR = plus Long Temps de fabrication Résiduel.

Contrairement à l'heuristique précédente, la pièce choisie est celle dont le temps résiduel est le plus long. Elle est utilisée avec succès dans les mêmes cas que LD, et lorsque dans l'ensemble des pièces, il s'en trouve quelques unes dont les temps de fabrication sont très supérieurs aux autres.

- Heuristique PNR.

PNR = plus Petit Nombre de tâches Résiduelles

La sélection porte sur le nombre de tâches restant à effectuer. Elle peut être intéressante lorsque les longueurs des gammes, évaluées en nombre d'opérations, sont très différentes ; ce n'est pas le cas des exemples numériques que nous avons testés.

- Heuristique GNR.

GNR = plus Grand Nombre de tâches Résiduelles.

La remarque faite à propos de l'heuristique précédente est encore valable.

- Heuristique CTR-D.

CTR-D = plus Court Temps Résiduel diminué de la Durée de la tâche à effectuer

A chaque pièce est affecté son temps résiduel diminué du temps de passage sur la machine devant laquelle elle est en attente. La méthode est très voisine de CTR et fournit des résultats comparables.

- Heuristique LTR-D.

LTR-D = plus Long Temps Résiduel diminué de la Durée de la tâche à effectuer.

La méthode est identique à LTR, à peu de choses près, et recouvre les mêmes cas d'application.

- Heuristique PR-D/TR.

PR-D/TR = plus Petit Ratio : Durée/Temps Résiduel.

Le critère de choix fait intervenir le rapport de deux grandeurs caractéristiques déjà rencontrées.

- Heuristique GR-D/TR.

GR-D/TR = plus Grand Ratio : Durée/Temps Résiduel.

La grandeur caractéristique est la même que celle de la méthode précédente. Seul le procédé de sélection diffère.

- Heuristique PAPS.

PAPS = Premier Arrivé Premier Servi.

Cette heuristique est très classique dans les problèmes de file d'attente ; la pièce sélectionnée est celle qui est arrivée la première en en-cours d'entrée de la machine.

1.3. EVALUATION DES PERFORMANCES.

Les performances d'une méthode sont évaluées d'après deux critères :

- la qualité de la solution trouvée : la méthode est d'autant plus performante que le résultat est proche de l'optimal.

- Le temps de résolution puisque nous sommes placés dans un environnement temps réel.

Il nous faut donc examiner successivement ces deux points.

Qualité de la solution.

Les valeurs numériques des solutions apportées par les heuristiques non convergentes sont données dans le tableau ci-dessous.

Problèmes / Méthodes	6 x 6	10 x 10	20 x 5
CD	88	1074	1267
LD	77	1305	1581
CTR	70	1334	1392
LTR	61	1110	1501
PNR	70	1332	1461
GNR	68	1224	1607
CTR-D	70	1334	1451
LTR-D	60	1092	1454
PR-D/TR	68	1017	1259
GR-D/TR	72	1301	1670
PAPS	65	1172	1645

Pour pouvoir juger de la qualité de ces résultats, il faut disposer d'une base de comparaison. Lorsque l'on connaît la solution optimale, il est possible de les situer par rapport à elle. Dans le cas contraire, il est intéressant de les comparer à un échantillon de solutions aléatoires qui est caractérisé par ses valeurs minimale, maximale et moyenne. La génération de solutions aléatoires de problèmes d'atelier est très simple : chaque fois qu'une décision est à prendre, la pièce à faire passer est choisie au hasard parmi celles qui sont dans l'en-cours d'entrée de la machine considérée.

Le tirage d'un échantillon de taille 100 a donné les résultats suivants :

Problème / Valeur	6 x 6	10 x 10	20 x 5
MINIMALE	58	1089	1346
MAXIMALE	87	1407	1683
MOYENNE	67	1229	1520

L'examen de ces deux tableaux conduit à un certain nombre de constatations :

- A l'exception de LD et GR-D/TR, toutes les heuristiques ont fourni dans au moins un des trois cas une solution meilleure que l'espérance des solutions aléatoires. Elles sont donc "optimisantes" au sens que leur utilisation est préférable à la génération au hasard. Ceci est d'autant plus vrai que le temps de calcul nécessaire à la mise en oeuvre de l'une d'entre elles est très voisin de celui qu'il faut pour construire une solution aléatoire.

  Cependant, il n'y a que deux méthodes qui soient meilleures que l'aléatoire pour les trois cas étudiés. Ce sont LTR et LTR-D dont les critères de choix sont pratiquement équivalents.

- Une observation plus fine des résultats pousse à faire une distinction entre problèmes carrés, dont le nombre de pièces n'est que peu supérieur à celui des machines (6 x 6 et 10 x 10), et problèmes rectangulaires (20 x 5). Dans ce dernier cas, les heuristiques de sélection sur les grandeurs caractéristiques minimales (CD, CTR, PNR, CTR-D, PR-D/TR) ont toujours donné des résultats meilleurs que leurs opposées respectives (LD,LTR,GNR, LTR-D, GR-D/TR). Pour les problèmes carrés au contraire, elles ont été d'une manière générale moins performantes (7 fois sur 10).

- La constatation précédente est infirmée par deux exceptions remarquables puisque ce sont CD et PR-D/TR qui ont fourni au problème 10 x 10 les meilleures des solutions trouvées, alors que celles qu'elles ont apportées au problème 6 x 6 se sont révélées être parmi les plus mauvaises ; le coût produit par CD est même supérieur à la valeur maximale de l'échantillon aléatoire. Ce qui tend à prouver que l'adaptation d'une méthode à un problème n'est pas fonction uniquement de sa structure carrée ou rectangulaire, mais fait intervenir d'autres facteurs plus complexes tels que le type des gammes de fabrication.

Temps de calcul.

Les temps de calcul nécessaires à la mise en oeuvre des heuristiques proposées sont très voisins les uns des autres ; ceci n'a rien d'étonnant dans la mesure où leurs fonctionnements sont identiques : à chaque fois qu'une décision est à prendre, on cherche la plus grande ou la plus petite des grandeurs caractéristiques des pièces disponibles. PAPS fait exception à cette règle puisque la pièce est choisie directement sans qu'il soit besoin d'effectuer une comparaison entre ces grandeurs. Les valeurs des temps observés (1) en secondes sont les suivantes (PAPS excepté) :

Problème / Temps	6 x 6	10 x 10	20 x 5
MINIMAL	0,53	1,66	2,28
MAXIMAL	0,59	1,74	2,50
MOYEN	0,55	1,70	2,37

Il faut s'attendre à ce que ces temps soient proportionnels au nombre n.m de décisions à prendre et au nombre moyen de pièces en en-cours d'entrée. Celui-ci dépend de l'allure des gammes et en particulier de la présence de goulots d'étranglement. On peut l'approcher par l'expression $(\frac{n}{m})^k$. Pour les problèmes testés une bonne évaluation de k est $\frac{1}{4}$ et dans ces conditions, les temps de calcul en secondes sont donnés par l'expression :

---

(1) Tous les algorithmes étudiés ont été programmés sur l'ordinateur Philips P1175 de l'Ecole Nationale Supérieure des Mines de Saint-Etienne.

$$T \simeq 1{,}6\ nm\left(\frac{n}{m}\right)^{\frac{1}{4}} 10^{-2}$$

Le temps de résolution d'un problème 20 x 10, peut être évalué à 4 secondes d'après cette formule. En admettant pour le délai de calcul imposé la limite sévère d'une minute, on voit qu'une faible fraction de ce délai (1/15) est utilisée pour trouver la solution avec l'un quelconque des algorithmes proposés.

Si on rapproche cette remarque de celle qui a été faite à propos de la qualité des résultats, il semble qu'une bonne approche pour résoudre un problème donné consiste à lui appliquer successivement les quelques heuristiques qui se sont révélées les plus performantes pour des problèmes de même type, jusqu'à épuisement du délai imparti. Dans l'exemple choisi, on peut essayer au plus 15 méthodes différentes. On conserve comme solution finale la meilleure de celles qui ont été ainsi obtenues.

2 - HEURISTIQUES ITERABLES.

2.1. PRINCIPE.

Comme les précédentes, ces méthodes ne permettent de construire qu'une seule solution réalisable. Mais la qualité de celle-ci est maintenant fonction de l'initialisation de l'algorithme, et non plus seulement des données caractéristiques du problème. Il est donc possible de les itérer plusieurs fois, à partir d'initialisations différentes (choisies par exemple de manière aléatoire).

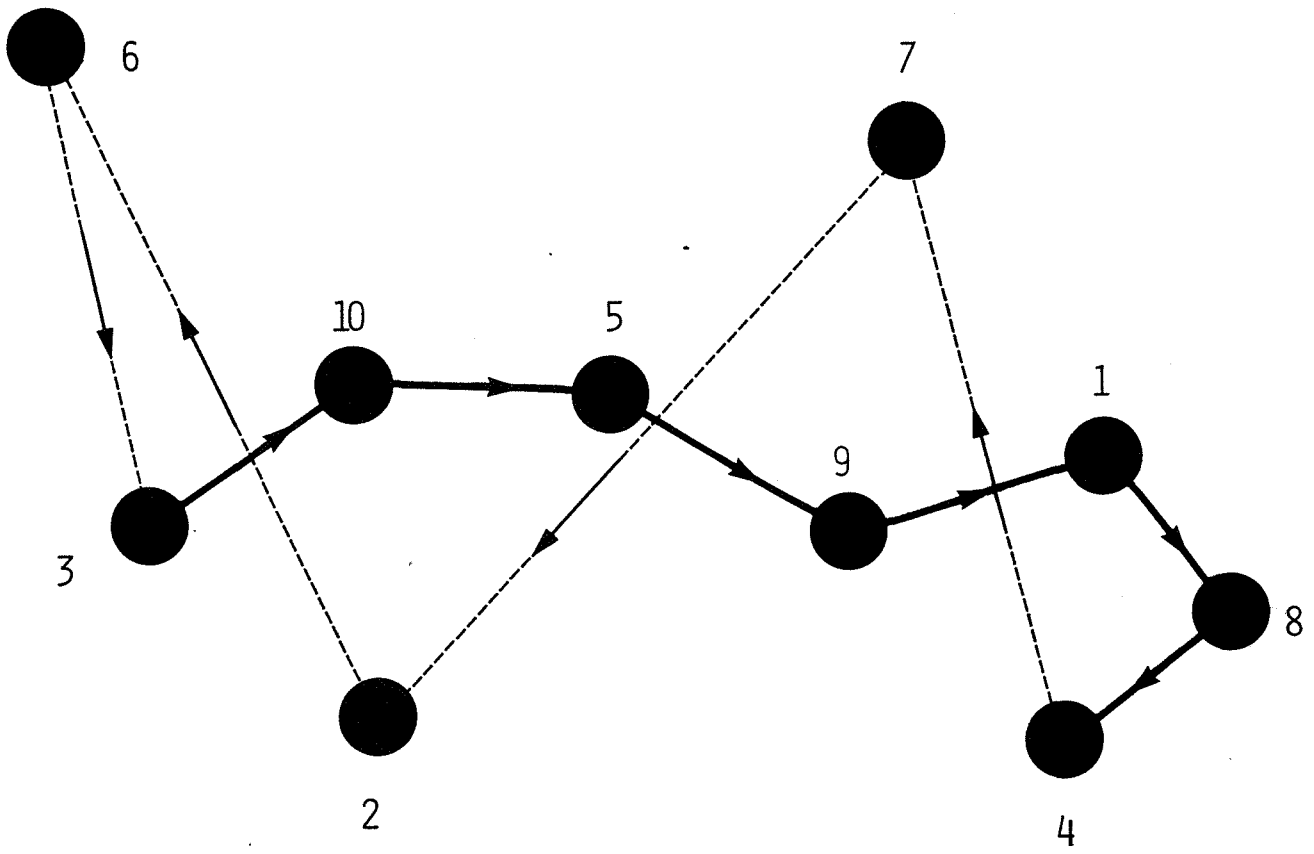
2.2. APPLICATION AUX PROBLEMES D'ORDONNANCEMENT.

Les méthodes sont testées sur les problèmes de voyageur de commerce ; il s'agit de trouver un circuit hamiltonien, c'est-à-dire qui passe une fois et une seule par chacune desn villes, de longueur minimale. Le problème est caractérisé par la matrice des distances séparant ces villes.

Pour chacun des trois exercices résolus, cette matrice présente la particularité d'être symétrique. Deux heuristiques sont fréquemment utilisées.

2.2.1. Algorithme de la ville la plus proche.

La première de ces méthodes est inspirée du comportement naturel du voyageur : lorsqu'il se trouve dans une ville, il cherche la plus proche de celles qu'il lui reste à visiter, et s'y rend. Le processus est itéré jusqu'à épuisement des villes. Dans certains cas, cet algorithme donne de très bons résultats : c'est particulièrement vrai lorsque les villes à visiter sont réparties à proximité des côtés d'un polygone convexe. Il est beaucoup moins performant lorsqu'elles se trouvent disséminées sur la totalité de la surface délimitée par le polygone enveloppant : en effet, la visite de certaines villes qui a été négligée lorsque le voyageur s'en trouvait à proximité est remise en fin de parcours.(cas des villes 2, 6, 7 sur la figure).



Algorithme de la ville la plus proche (départ de la ville 3)

L'initialisation de la méthode suppose que l'on fixe le point de départ. Il y a donc autant de solutions différentes que de villes. Dans le cas d'une matrice non symétrique, on peut également fixer le point d'arrivée : le nombre des solutions différentes est alors 2n. Lorsque le temps de calcul alloué le permet, il y a tout intérêt à parcourir l'ensemble de ces solutions réalisables en itérant plusieurs fois la méthode pour ne conserver que la meilleure d'entre elles.

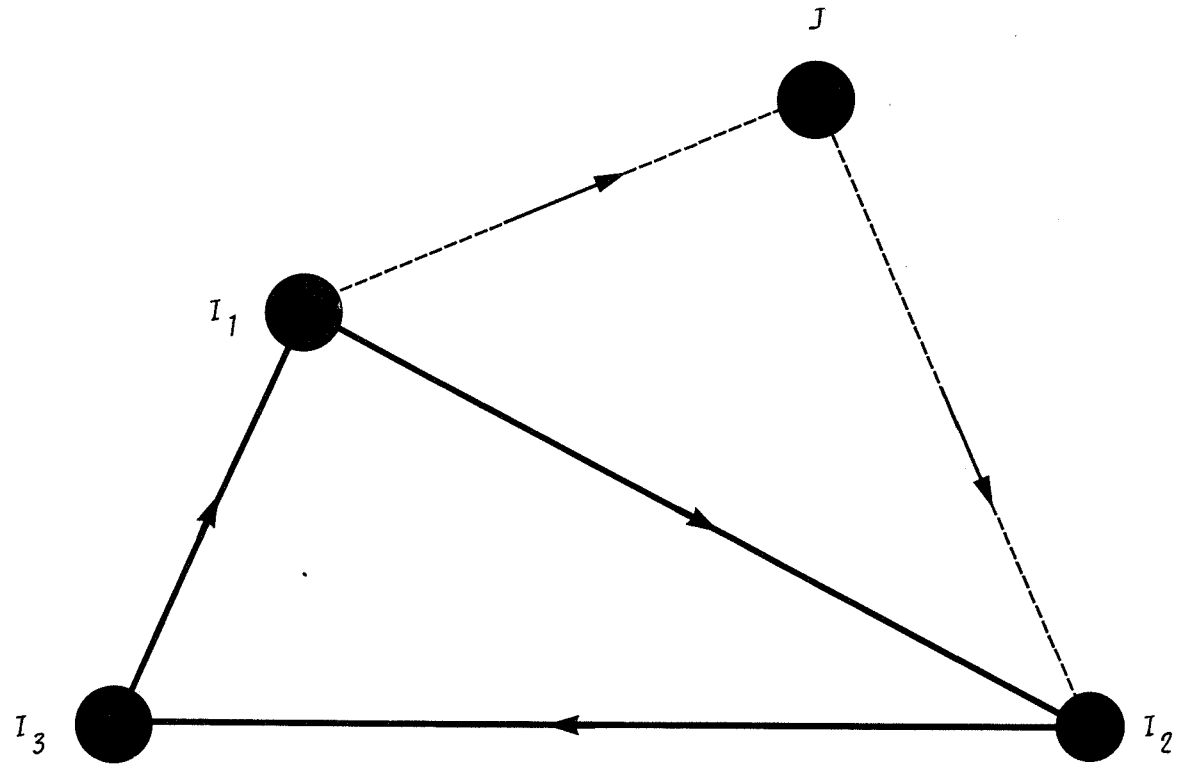
2.2.2. Algorithme de KARG [7].

La première phase de la méthode consiste à classer les villes de manière aléatoire. Une fois que cette opération est réalisée, on prend les trois premières villes déterminées par cet ordre pour construire un circuit élémentaire à trois sommets i_1, i_2 et i_3. Son coût C_3 est égal à la somme des longueurs des arcs constitutifs, c'est-à-dire :

$$C_3 = d_{i_1 i_2} + d_{i_2 i_3} + d_{i_3 i_1}$$

L'algorithme procède par agglomération successive des villes au circuit ; soit j la 4ème ville à visiter. Il existe trois possibilités pour l'intégrer au circuit déjà constitué :

- ou bien on la place entre i_1 et i_2 (cf. figure).
- ou bien entre i_2 et i_3
- ou bien enfin entre i_3 et i_1.



Algorithme de KARG - Insertion de la ville j.

Dans tous les cas, l'insertion de cette nouvelle ville conduit à supprimer l'un des arcs existants et à en ajouter deux nouveaux, et l'on a :

$$C_4 = C_3 + (d_{i_k j} + d_{j\, i_{k+1}} - d_{i_k i_{k+1}})$$

où k+1 = 1 si k=3

Il suffit donc de trouver l'indice k qui minimise l'expression entre parenthèses pour que l'insertion de j soit réalisée au moindre coût. On a ainsi déterminé un nouveau circuit à 4 sommets et il suffit de recommencer l'opération avec une nouvelle ville jusqu'à ce qu'ellesaient toutes été agglomérées. A chaque itération, le nombre de positions à tester augmente d'une unité.

La qualité de la solution finale dépend bien évidemment de l'ordre initial de classement des villes. Dans quelques cas particuliers, il existe des méthodes de détermination de cet ordre qui assurent l'optimalité du résultat trouvé [7]. Elles supposent que les villes sont situées dans un plan et font intervenir les angles formés par les droites générées par les différents couples de villes. Nous n'en parlerons pas ici car elles ne peuvent pas s'appliquer à l'ensemble des problèmes d'ordonnancement formulables en termes de voyageur de commerce,et nous nous contenterons donc de générer l'ordre de prise en compte de manière aléatoire.

Il faut enfin remarquer que lorsque la matrice des distances n'est pas symétrique, la seule modification à apporter concerne le circuit initial qui comporte non plus trois, mais deux sommets.

2.3. EVALUATION DES PERFORMANCES.

qualité de la solution.

Pour un problème à n variables, on a itéré n fois l'algorithme de la ville la plus proche en prenant successivement chaque ville comme point de départ, et 100 fois la méthode de KARG à partir d'ordres aléatoires différents. Pour chaque série d'essais, on donne les valeurs minimales, maximales et moyennes des solutions trouvées.

METHODES	Problèmes / Valeurs	33	42	57
VILLE LA PLUS PROCHE	MINIMALE	11 426	864	14 411
	MAXIMALE	14 228	1015	18 518
	MOYENNE	12 523	927	15 916
KARG	MINIMALE	10 925	699	13 132
	MAXIMALE	12 815	820	15 563
	MOYENNE	11 591	754	14 417

Ces valeurs sont à comparer à la fois :

- aux solutions optimales qui ont été calculées pour les deux premiers problèmes et à la meilleure solution connue du dernier,

- aux espérances mathématiques des solutions aléatoires.

Celles-ci ont été évaluées rigoureusement de la manière suivante : Compte tenu du fait que tout circuit comporte n arcs et que chacun d'entre eux est présent avec la même fréquence dans l'ensemble de toutes les solutions possibles, il suffit de multiplier par n le coût moyen d'un arc pour obtenir l'espérance cherchée.

Problèmes / valeurs	33	42	57
OPTIMALE	10 861	699	12 955
ESPERANCE	42 331	3 110	59 882

L'examen de ces tableaux suggère les remarques suivantes :

- l'utilisation de l'une ou l'autre de ces heuristiques est beaucoup plus efficace que la génération aléatoire.

- la deuxième méthode s'est toujours avérée plus performante que la première : les valeurs maximales trouvées par l'algorithme de KARG sont presque toutes inférieures aux valeurs moyennes de la première méthode : pour un seul problème, elle lui est très légèrement supérieure. On peut même constater que lorsque n = 42, la plus mauvaise solution fournie par KARG est inférieure à la meilleure des solutions produites par l'autre méthode.

- enfin, l'algorithme de KARG peut être considéré comme très optimisant. Sa mise en oeuvre a conduit une fois à l'optimum. De plus, les écarts entre les valeurs moyennes et les valeurs optimales, exprimées en pourcentage, n'ont jamais été supérieures à 12 %.

Temps de calcul.

Les temps de calcul moyens nécessaires pour trouver une solution en appliquant chacune des deux méthodes sont les suivants :

problème / méthode	33	42	57
VILLE LA PLUS PROCHE	0,85	1,32	2,41
KARG	0,57	0,88	1,55

On peut s'attendre à ce que ces temps soient proportionnels à n^2, en effet :

- pour trouver la ville qu'il n'a pas encore visitée, et qui est la plus proche de celle où il se trouve, le voyageur doit passer toutes les autres en revue. Ce processus est itéré n-1 fois : il y a donc $(n-1)^2 \simeq n^2$ comparaisons à faire.

- dans l'algorithme de KARG, il y a trois positions possibles pour la quatrième ville, quatre pour la cinquième, n-1 pour la $n^{ième}$. Le nombre de tests est alors :

$$\frac{n(n-1)}{2} - 4 \simeq n^2.$$

Dans la plage des valeurs de n considérées, les temps de calcul sont donnés par les expressions :

$T \simeq 7{,}5\ n^2\ 10^{-4}$	pour le premier algorithme
$T \simeq 5{,}0\ n^2\ 10^{-4}$	pour le second

La méthode de KARG est donc incontestablement la plus performante puisqu'elle conduit à des solutions de meilleure qualité en des temps de calcul plus courts.

3 - INTERET DE CES METHODES.

L'avantage essentiel que présentent les heuristiques non convergentes réside dans la rapidité de leur mise en oeuvre

Dans tous les cas que nous avons examinés, les temps de calcul sont de la forme :

$$T \simeq p^a$$

dans laquelle :

- p est le nombre de variablesdu problème : il est égal au nombre n.m de décisions s'il s'agit de l'ordonnancement d'un atelier, ou au nombre n de villes dans le cas d'un problème de voyageur de commerce.

- a est un entier inférieur ou égal à 2.

Or, le temps de l'algorithme le plus rapide que l'on puisse imaginer, qui consiste à attribuer une valeur aléatoire à chacune de ces p variables, est au moins proportionnel à p. En outre, si nous prenons comme méthode de référence celle qui consiste à effectuer un tri des variables suivant un certain critère, son temps est en pLog p. Les heuristiques testées sont donc pratiquement aussi rapides, à un coefficient de proportionnalité près.

En ce qui concerne la qualité de la solution, il est impossible de tirer des conclusions générales ; si certaines méthodes s'avèrent très performantes (KARG), les résultats fournis sont souvent imprévisibles car très dépendants de la structure particulière du problème traité.

Mais, ce n'est pas là que nous semble résider l'inconvénient majeur de ce type d'approches, mais plutôt dans le principe même de la non convergence . A supposer en effet que le temps de résolution imparti soit de l'ordre de la minute, et puisque leur application ne nécessite que quelques secondes, on ne peut pas toujours tirer profit du temps résiduel pour améliorer la solution obtenue : le plus souvent, la méthode ne peut être appliquée qu'une fois (heuristique d'atelier) ou un nombre fini de fois (ville la plus proche) ; l'algorithme de KARG fait exception dans la mesure où il peut être itéré autant qu'on le désire, à partir d'ordres initiaux aléatoires, mais alors les solutions construites sont indépendantes entre elles. C'est là la différence caractéristique entre les heuristiques non convergentes et convergentes que nous allons maintenant présenter.

CHAPITRE 3

LES HEURISTIQUES CONVERGENTES

CHAPITRE III

LES HEURISTIQUES CONVERGENTES

Le principe général de ces méthodes consiste à explorer un sous-ensemble plus ou moins vaste des solutions réalisables. Une telle exploration peut être faite de deux manières ; elle est :

- soit aléatoire lorsque les solutions examinées sont indépendantes les unes des autres ; c'est le cas lorsque l'on *itère* un certain nombre de fois une méthode de résolution à partir d'initialisations aléatoires. Aussi l'algorithme de la ville la plus proche et celui de KARG peuvent-ils être itérés respectivement n et $\frac{n!}{2}$ fois, si la matrice des distances est symétrique, et conduire à autant de solutions indépendantes.

- soit dirigée : la méthode est mise en oeuvre *une seule* fois et conduit à l'examen d'un certain sous-ensemble de solutions réalisables qui est construit d'après des règles qui lui sont propres. Ce dernier type d'exploration est caractéristique des heuristiques convergentes.

Il est clair que la qualité de la solution obtenue et le temps de résolution dépendent essentiellement de la manière dont est conduite l'exploration. Si elle est réalisée de telle sorte qu'on soit certain d'obtenir la solution optimale, la méthode peut être qualifiée de globale et non pas d'heuristique. Mais les temps de calcul nécessaires sont alors considérables et incompatibles avec la contrainte temps réel qui est imposée. C'est pourquoi il ne nous est pas possible de laisser la procédure se dérouler jusqu'à son terme. Elle sera donc arrêtée au bout d'un certain temps et la meilleure solution connue à cet instant sera retenue comme solution du problème. C'est la raison pour laquelle la méthode,

quoique globale dans son principe, est classée parmi les heuristiques convergentes. Dans d'autres cas, l'exploration conduit à une convergence vers une solution qui n'est pas nécessairement l'optimale. Le nom d'heuristique est alors pleinement justifié.

Deux de ces heuristiques seront successivement étudiées ; il s'agit des méthodes de recherche en arbre et de leurs aménagements possibles pour tenir compte de la contrainte temps réel, et de l'algorithme des distances. Elles seront exclusivement testées sur les problèmes de voyageur.

1 - METHODES DE RECHERCHE EN ARBRE.

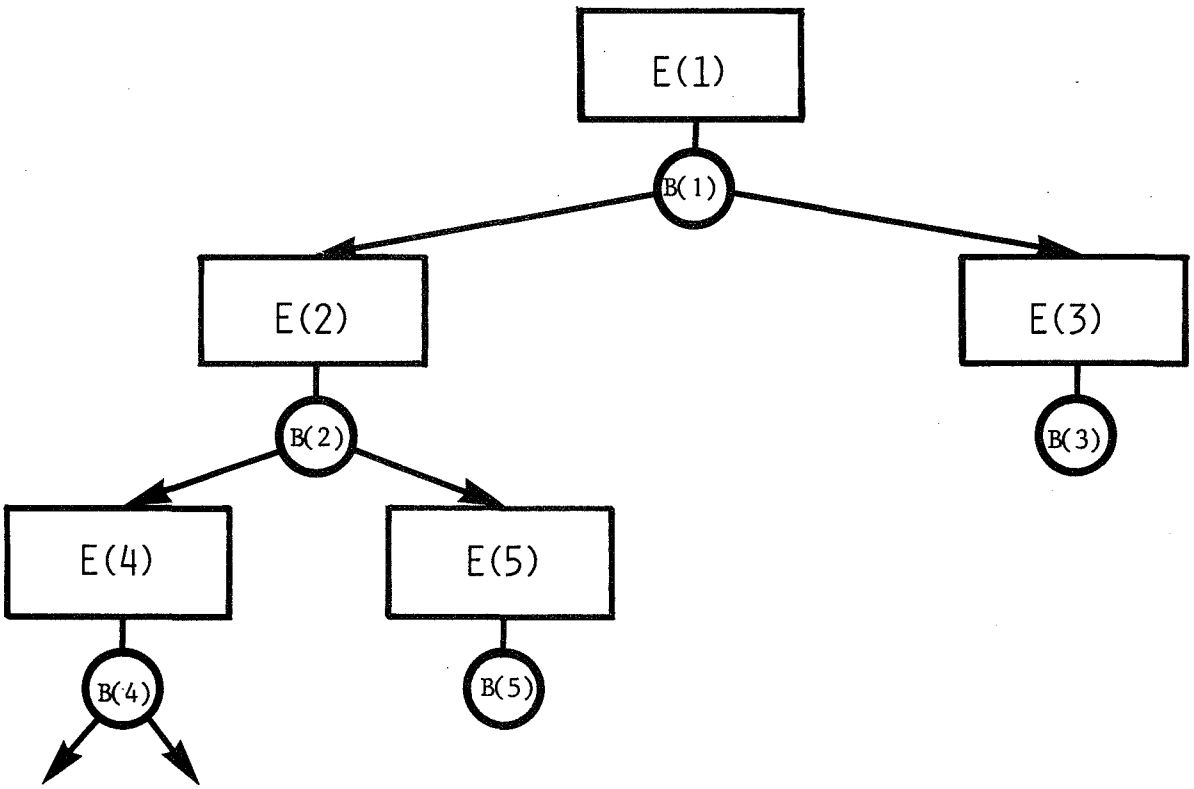
Les méthodes de recherche en arbre sont très classiques en matière de résolution de problèmes combinatoires. On les connaît également sous le nom de Séparation et Evaluation Progressive (SEP) ou Branch and Bound.

1.1. PRINCIPE.

Il consiste en la construction d'une arborescence de segmentation de l'espace des solutions réalisables. Pour clarifier l'exposé, nous supposerons que le critère consiste à minimiser une certaine grandeur, et nous appellerons "solution standard" la meilleure solution connue à chaque instant. Si aucune solution réalisable n'est donnée à priori, le coût standard C est initialisé à une valeur très grande.

La mise en oeuvre de la méthode est alors la suivante : l'ensemble des solutions réalisables E(1) du problème est décomposé en deux sous-ensembles disjoints et complémentaires : E(2) et E(3).

Pour chacun d'entre eux, on calcule une borne inférieure (respectivement B(2) et B(3)) de telle sorte que les valeurs du critère pour toutes les solutions qui en sont issues soient certainement supérieures ou égales à cette borne. On choisit ensuite un nouveau sous-ensemble à partir duquel on peut continuer le processus de décomposition : ce peut être par exemple celui dont la borne inférieure est minimale.



Construction des premiers noeuds de l'arborescence

Ces décompositions successives conduisent à construire des sous-ensembles de taille de plus en plus réduite, et on aboutit finalement à des feuilles terminales de l'arbre de segmentation ne contenant qu'une seule solution réalisable pour laquelle la borne associée est égale à la valeur exacte du critère. On met à jour, s'il y a lieu, le coût standard C, et on reprend l'exploration des sous-ensembles qui ont été laissés de côté, à condition que leur borne soit inférieure à C.

On continue ainsi le processus de branchement et de séparation jusqu'à obtenir un coût standard inférieur ou égal à toutes les bornes des sous-ensembles non examinés. La solution standard constitue alors l'optimum cherché.

L'algorithme général de recherche en arbre est de la forme ci-contre. Sa mise en oeuvre nécessite la définition de trois règles :

1 - La règle 1 de calcul des bornes inférieures (Etapes 1 et 3).

2 - La règle 2 de segmentation des sous-ensembles (Etape 3).

3 - La règle 3 de branchement, c'est-à-dire de choix du sous-ensemble à partir duquel on va continuer l'exploration.

ETAPE 1 : *début*

C ← + ∞
E(1) ← Ensemble des solutions réalisables du problème.
B(1) ← Borne inférieure de E(1)

ETAPE 2 : *Si* tous les sous-ensembles E(i) ont été examinés,

alors fin
sinon choisir un sous-ensemble E(j) non exploré
Si B(j) ≥ C *alors aller à* Etape 2.
Sinon aller à Etape 3.

ETAPE 3 : *Si* E(j) ne contient qu'une solution réalisable,

alors C ← min (C, B(j)) *aller à* Etape 2
sinon segmenter E(j) en deux sous-ensembles E(2j) et E(2j + 1)
B(2j) ← Borne inférieure de E(2j)
B(2j + 1) ← Borne inférieure de E(2j + 1)
aller à Etape 2.

Algorithme général de recherche en arbre

La règle 1 est la plus importante : les performances de l'algorithme dépendent essentiellement du calcul des bornes.

La règle 2 est telle que sa définition dépend en grande partie de la nature du problème à traiter.

La règle 3 conduit à envisager deux approches de branchement à partir :

- soit du sous-ensemble dont la borne inférieure est minimale.
- soit du sous-ensemble le plus récemment créé.

L'application de cette deuxième approche fait qu'en général, les opérations de branchement sont plus nombreuses que si l'on utilisait la première, mais elle nécessite moins de place mémoire.

1.2. APPLICATION AUX PROBLEMES D'ORDONNANCEMENT.

Divers auteurs ont tenté de résoudre le problème d'atelier à n pièces et m machines par des algorithmes de recherche en arbre (voir en particulier [1] et [3]). Il semble pourtant qu'ils ne soient pas suffisamment performants pour y apporter en temps réel une solution satisfaisante dès lors que n et m sont grands. C'est pourquoi, nous restreindrons notre étude aux problèmes de voyageur. Plusieurs variantes ont été testées qui ne diffèrent que par le calcul des bornes inférieures. La règle de segmentation est toujours la même ainsi que la règle de branchement : compte tenu de la taille importante des problèmes à traiter, nous avons choisi la deuxième approche de cette règle pour minimiser la place mémoire nécessaire.

1.2.1. Calcul sur les coûts réels.

Une solution est déterminée par la sélection de n arcs. Nous allons construire une arborescence en associant à chaque sommet :

- r arcs dont la candidature a été examinée et qui ont été retenus.
- s arcs dont la candidature a été examinée et qui ont été rejetés.
- n-r arcs candidats à la solution du problème, tous compatibles avec les arcs retenus (pas de formation de sous-circuits).
- une valeur de la borne inférieure pour les solutions issues de ce sommet. Cette valeur est obtenue par sommation des coûts des arcs retenus et candidats.

Les arcs examinés définissent le sous-ensemble des solutions réalisables issues du sommet considéré : dans toutes ces solutions sont présents les arcs retenus et absents les arcs rejetés.

Afin que la borne inférieure soit non décroissante, le long d'un chemin de l'arbre, il faut définir une règle de sélection des arcs candidats. Celle-ci se fait de manière très simple dans l'ordre de leurs coûts croissants.

Au départ, il n'y a aucun arc retenu puisqu'aucun examen de candidature n'a eu lieu. Par contre, il y a des candidats au nombre de n.

Ces candidats sont les n premiers arcs de la liste triée. La valeur de la borne inférieure associée est égale à la somme des coûts affectés à ces arcs.

Considérons maintenant un noeud quelconque de l'arborescence et examinons le premier arc de la liste des candidats. Cet examen conduit à la construction de deux nouveaux noeuds :

- Noeud 1 : Le candidat est accepté et passe dans la liste des retenus dont le nombre devient :

$$r \leftarrow r + 1$$

Il faut alors s'assurer que les candidats sont toujours compatibles avec la nouvelle liste des retenus. Deux éventualités peuvent se produire :

- on trouve n-r candidats compatibles. La valeur de la borne inférieure est alors calculée par sommation des coûts des arcs retenus et candidats.

- il n'y a pas suffisamment de candidats compatibles parmi les arcs pour lesquels aucune décision n'a été prise. Cela signifie que le sous-ensemble relatif au sommet considéré est vide. La valeur de la borne inférieureest alors mise à l'infini.

- Noeud 2 : Le candidat est rejeté. La mise à jour de la nouvelle liste des candidats et le calcul de la borne inférieure sont effectués comme précédemment.

Exemple :

Le problème à résoudre comporte 6 villes. Il faut donc sélectionner 6 arcs. Ceux-ci sont numérotés dans l'ordre des coûts croissants. On suppose la matrice symétrique. Le tableau des arcs est le suivant :

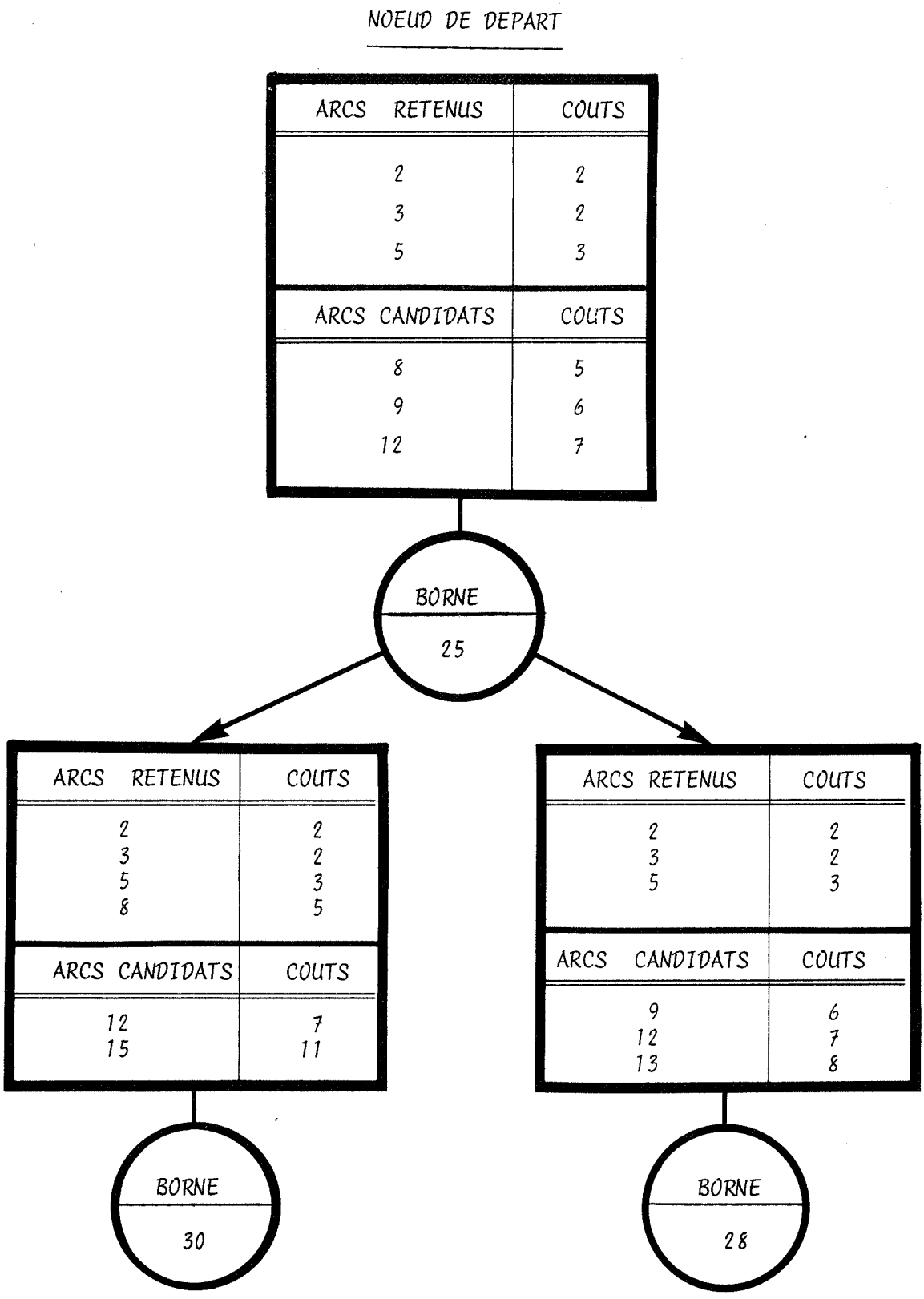
NUMERO	SOMMETS RELIES		COUT
1	4	5	1
2	2	4	2
3	5	6	2
4	2	5	3
5	1	5	3
6	1	2	4
7	3	4	5
8	4	6	5
9	1	4	6
10	3	5	6
11	1	6	7
12	2	3	7
13	3	6	8
14	2	6	8
15	1	3	11

Plaçons nous en un noeud où 3 arcs ont déjà été retenus (2,3,5). Les candidats compatibles sont au nombre de 3 (8, 9, 12) et la borne inférieure est égale à la somme des coûts de tous ces arcs (25).

Deux nouveaux noeuds sont créés :

- noeud 1 : l'arc 8 passe dans la liste des arcs retenus dont le nombre est maintenant égal à 4. Il faut donc trouver deux arcs candidats. Ceux-ci sont à choisir en priorité parmi ceux du noeud de départ, en éliminant ceux que la sélection de l'arc 8 a rendu incompatibles avec la nouvelle liste des arcs retenus. Puis on complète, s'il y a lieu, la liste des candidats. Dans l'exemple proposé, l'arc 9 a été éliminé et remplacé par l'arc 15. La borne inférieure est alors égale à 30.

- noeud 2 : l'arc 8 est rejeté. Il doit être remplacé par un nouveau candidat compatible (13). La valeur de la borne est 28.


NOEUD DE DEPART
ARCS RETENUS
COUTS
2
2
3
2
5
3
ARCS CANDIDATS
COUTS
8
5
9
6
12
7
BORNE
25
ARCS RETENUS
COUTS
2
2
3
2
5
3
8
5
ARCS CANDIDATS
COUTS
12
7
15
11
BORNE
30
ARCS RETENUS
COUTS
2
2
3
2
5
3
ARCS CANDIDATS
COUTS
9
6
12
7
13
8
BORNE
28
NOEUD 1
NOEUD 2

Nous pouvons faire les remarques suivantes :

1 - Il y a toujours intérêt à développer l'arborescence verticalement et non horizontalement.
En effet, l'inscription des candidats dans l'ordre croissant de leurs coûts implique que la borne inférieure ne peut être que non décroissante le long d'un chemin de l'arborescence ; d'où il résulte que si la borne en un noeud est supérieure à la valeur du critère en une feuille terminale, il est inutile de continuer l'exploration du sous-ensemble des solutions issues de ce noeud. On effectue alors une "coupure" de la branche en ce point.

2 - On atteint une feuille terminale, c'est-à-dire une solution réalisable lorsque les arcs retenus sont au nombre de n. On procède s'il y a lieu à la mise à jour de la solution standard et, pour des raisons de place mémoire, on poursuit l'exploration à partir du premier ancêtre de la feuille dont la valeur de la borne est inférieure au coût standard (l'arbre est visité en préordre).

L'algorithme tel qu'il est proposé conduit à la détection de la solution optimale à condition de disposer d'un temps de calcul suffisant. Bien que toutes les solutions réalisables ne soient pas énumérées puisque l'on pratique des coupures dans l'arbre de recherche, ce temps varie en principe exponentiellement avec le nombre de variables du problème. Il est donc hors de question de laisser la procédure se dérouler jusqu'à son terme si n est grand.

Dans la pratique, elle est arrêtée dès que l'on a épuisé le délai de calcul imparti (qui peut être de l'ordre de quèlques secondes ou quelques minutes selon les cas).

Les essais que nous avons effectués nous ont permis de constater que les solutions réalisables construites ne diffèrent que par quelques arcs qui sont les derniers sélectionnés puisque l'arbre est parcouru en préordre (ce nombre est compris entre 10 et 20 pour un temps de calcul d'une dizaine de minutes). Ceci veut dire que le choix des premiers arcs retenus a une influence considérable puisque leur sélection ne sera jamais remise en cause.

Or, le critère de choix des candidats dans l'ordre des coûts croissants n'est peut être pas le meilleur. C'est pourquoi, on a essayé de caractériser un arc par une grandeur qui mesure plus exactement que son coût l'intérêt qu'il y a à le sélectionner.

1.2.2. - Calcul sur les coûts transformés.

Soit à trouver un circuit hamiltonien minimal dans un graphe complet d'ordre n, c'est-à-dire, que de toute ville i on puisse se rendre à n'importe quelle ville $j \neq i$. La matrice P des distances p_{ij} est pour le moment supposée non symétrique.

Calculons :

- le nombre de circuits hamiltoniens différents :

$$(n - 1)!$$

- le nombre de circuits hamiltoniens contenant l'arc (k, l) : étant donné qu'un circuit comporte n arcs, que le nombre d'arcs différents est n(n-1) et que chaque arc est présent avec la même fréquence dans l'ensemble des circuits, le nombre cherché est égal à :

$$\frac{1}{n(n-1)} \times n \times (n-1)! = (n-2)!$$

- le nombre de circuits ne contenant pas l'arc (k,l) :

$$(n-1)! - (n-2)! = (n-2)(n-2)!$$

Posons :

$S = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij}$ somme des éléments de la matrice P.

$S_1(k) = \sum_{j=1}^{n} p_{kj}$ somme des éléments de la ligne k de la matrice P.

$S_2(1) = \sum_{i=1}^{n} p_{il}$ somme des éléments de la colonne 1 de la matrice P.

Nous pouvons également évaluer :

- le coût moyen de tous les circuits hamiltoniens :

$$A = \frac{S}{n(n-1)} \times n = \frac{S}{n-1}$$

- le coût moyen des circuits contenant l'arc (k,l) :
  Ne peuvent participer à ces circuits :
  - . l'arc (l,k)
  - . les arcs issus de k sauf (k,l)
  - . les arcs aboutissant à l sauf (k,l)

L'arc (k,l) est présent dans tous ces circuits, les autres y sont présents avec la même fréquence. Ils sont au nombre de (n-1)(n-2). Le coût moyen B cherché est donc :

$$B = p_{kl} + \frac{S - S_1(k) - S_2(l) - p_{lk} + p_{kl}}{(n-1)(n-2)} \times (n-1)$$

$$= p_{kl} + \frac{1}{n-2} (S - S_1(k) - S_2(l) - p_{lk} + p_{kl})$$

- le coût moyen des circuits hamiltoniens ne contenant pas l'arc (k,l) :
  un calcul de moyenne pondérée donne :

$$C = \frac{1}{(n-2)^2} \quad [(n-3)S + S_1(k) + S_2(l) - (n-1)p_{kl} + p_{lk}]$$

Il est alors possible de caractériser l'arc(k,l) par l'une des trois mesures de préférence :

1 - <u>Coût moyen des circuits hamiltoniens dont il fait partie</u> :

$$U_{kl} = B = p_{kl} + \frac{1}{n-2} (S - S_1(k) - S_2(l) - p_{lk} + p_{kl})$$

Si on simplifie ce coût en négligeant les facteurs constants, il vient :

$$U_{kl} = (n-1)p_{kl} - S_1(k) - S_2(l) - p_{lk}$$

2 - Opposé du coût moyen des circuits hamiltoniens dont il ne fait pas partie.

$$V_{kl} = - C = - \frac{1}{(n-2)^2} \left[(n-3)\ S + S_1(k) + S_2(l) - (n-1)p_{kl} + p_{lk} \right]$$

Par simplification on trouve :

$$V_{kl} = (n-1)p_{kl} - S_1(k) - S_2(l) - p_{lk}$$

3 - Différence entre les deux coûts précédents :

$$W_{kl} = B - C = \frac{1}{(n-2)^2} \left[(n-1)^2\ p_{kl} + S - (n-1)(S_1(k) + S_2(l) + p_{lk}) \right]$$

Par simplification on a :

$$W_{kl} = (n-1)\ p_{kl} - S_1(k) - S_2(l) - p_{lk}$$

Ces trois mesures sont donc identiques. Les calculs précédents restent bien entendu valables dans le cas d'une matrice symétrique.

Le problème peut donc être résolu en remplaçant le coût réel des arcs par cette valeur commune, ou coût transformé, et en appliquant la méthode de recherche en arbre telle qu'elle a été décrite précédemment. Il est clair que les valeurs des bornes inférieures et du critère ne seront plus les valeurs réelles de ces grandeurs aux noeuds et aux feuilles correspondantes et que donc la convergence n'est plus assurée vers la solution optimale, mais seulement vers une bonne solution. Cependant, cette transformation présente un avantage d'abord parce que les arcs sont caractérisés par une grandeur qui mesure plus fidèlement l'intérêt qu'il y a à les sélectionner, et ensuite parce que les coûts transformés s'étendent sur une plage de valeurs en général plus vaste que les coûts réels et qu'alors la convergence est plus rapide.

1.2.3. - Algorithme de LITTLE, MURTY, SWEENEY, KAREL [10]

L'intérêt de cette méthode réside dans la règle de calcul de la borne inférieure d'un sous-ensemble de solutions.

Supposons que nous ayons à trouver un circuit hamiltonien minimal dans une matrice non symétrique d'ordre n. Cela revient à dire qu'il nous faut sélectionner n éléments de cette matrice, un dans chaque ligne et dans chaque colonne, de manière à ne pas créer de boucles. Le coût de la solution est alors la somme des n éléments sélectionnés.

Il est alors possible de réduire la matrice en soustrayant de chaque ligne i son plus petit élément a_i de manière à faire apparaître un certain nombre de zéros, et en recommançant la même opération sur les colonnes dont nous désignerons par b_j le plus petit élément. Dans ces conditions, l'expression :

$$I = \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j$$

constitue une borne inférieure du coût de l'ensemble des circuits hamiltoniens. Pour construire une bonne solution, nous avons tout intérêt à sélectionner le plus grand nombre de zéros dans la matrice ainsi transformée.

Partant de là, montrons comment on peut évaluer la borne inférieure d'un sous-ensemble de solutions. Nous avons vu qu'un tel sous-ensemble est défini par un certain nombre d'arcs retenus et rejetés. Plaçons nous en un noeud de l'arborescence et examinons la candidature d'un nouvel arc (k,l). Deux noeuds sont créés :

- Noeud 1 : l'arc (k,l) est retenu ; pour toutes les solutions appartenant au nouveau sous-ensemble ainsi créé sont désormais interdites les sélections d'arcs commençant à la ville k ou aboutissant à la ville l. Dans la matrice représentative, cela veut dire qu'il faut "barrer" la ligne k et la colonne l : pratiquement on affecte aux éléments correspondants des coûts infinis. En outre, il y a encore un autre arc dont la sélection n'est plus possible. Deux cas sont à envisager :

  . ou bien l'arc (k,l)ne constitue par une chaîne avec les arcs déjà sélectionnés. L'arc interdit est alors (l,k)

  . ou bien l'arc (k,l) forme une chaîne avec les arcs déjà retenus. L'arc interdit dépend alors de ces derniers.

Exemple :

Supposons que les arcs (2,4), (5,7) et (4,3) aient été retenus. Ils constituent deux chaînes :

2-4-3 et 5-7

- si l'arc (6,1) est retenu, il ne forme aucune chaîne avec les précédents et l'arc interdit est (1,6).

- si l'arc (3,6) est retenu, il y a création d'une chaîne ;

2-4-3-6

et l'arc interdit est (6,2)

- si l'arc (7,2) est retenu, l'arc interdit et déterminé par la chaîne :

5-7-2-4-3

c'est donc (3,5).

Une fois que nous avons supprimé dans la matrice tous les arcs dont le choix est rendu impossible par celui de (k,l), il faut faire apparaître, s'il y a lieu, des zéros dans la nouvelle matrice réduite. La borne inférieure relative au noeud est alors égale à la somme de tous les nombres qui ont été soustraits aux lignes et aux colonnes. Il ne reste plus qu'à renouveler l'opération sur une matrice dont le rang a diminué d'une unité. Il peut arriver que dans une ligne ou une colonne non barrée tous les coûts soient infinis. Le sous-ensemble relatif au sommet associé est vide et sa borne inférieure est mise à l'infini.

- noeud 2 : l'arc (k,l) est rejeté ; on interdit sa sélection ultérieure en lui affectant un coût infini. On fait apparaître de nouveaux zéros dans la ligne et/ou sa colonne si nécessaire, et on calcule la borne inférieure correspondante. Le rang de la matrice demeure inchangé.

L'ordre dans lequel les arcs sont examinés influe fortement sur la rapidité de convergence de l'algorithme. L'arbre étant exploré dans le sens de la profondeur, on a tout intérêt à choisir les arcs de coût nul dans la matrice réduite dont la non sélection serait la plus coûteuse. L'heuristique de choix des arcs consiste à attribuer un poids à chacun des zéros de la matrice en lui affectant la somme des plus petits éléments de leur ligne et de leur colonne (non compris le zéro considéré).

Ce poids représente l'élévation minimale de la borne inférieure consécutive à leur non-sélection. Le zéro choisi est celui dont le poids est le plus grand.

La valeur de la borne produite par cet algorithme est supérieure à celle que fournit la méthode de calcul sur les coûts réels pour des sous-ensembles identiques ; elle est donc plus efficace. Cependant, son temps d'obtention est beaucoup plus élevé puisqu'il faut effectuer, pour chaque sélection d'arc, un certain nombre de manipulations sur les lignes et les colonnes de la matrice. De plus, la méthode de LITTLE, très performante par rapport aux autres lorsque la matrice est non symétrique, l'est beaucoup moins dans le cas contraire car la symétrie n'apporte aucune simplification notable.

1.3. EVALUATION DES PERFORMANCES.

Les tests ont été effectués sur les problèmes de voyageur à 33, 42 et 57 variables. Dans aucun cas la recherche en arbre n'a pu aller jusqu'à son terme. On a donc relevé le coût standard à des intervalles de temps réguliers pour pouvoir comparer les méthodes entre elles.

	33 VILLES					
Temps de calcul / Méthodes	5"	10"	15"	1'	5'	10'
COUTS REELS	12585	12234	11666	11666	11467	11308
COUTS TRANSFORMES	11322	10989	10989	10989	10989	10989
LITTLE	-	-	13014	13014	12847	12847

	42 VILLES					
Temps de calcul / Méthodes	10"	15"	30"	1'	5'	10'
COUTS REELS	973	891	891	835	819	819
COUTS TRANSFORMES	724	701	701	701	701	701
LITTLE	-	-	871	779	779	779

	57 VILLES					
Temps de calcul / Méthodes	20"	25"	30"	1'30"	5'	10'
COUTS REELS	14583	14583	14228	14228	14228	14228
COUTS TRANSFORMES	14136	14136	14084	13990	13814	13802
LITTLE	-	-	-	13727	13655	13655

De l'examen de ces tableaux il résulte que :

- La qualité de la solution standard dépend principalement de celle de la première solution détectée. Ceci résulte du fait que le parcours de l'arbre est effectué en préordre et que donc on cherche à améliorer cette solution par substitution de ses derniers arcs sélectionnés. Le temps de calcul étant limité, il n'est pas possible de remonter très haut dans l'arbre. L'importance des premiers arcs sélectionnés est alors considérable puisqu'ils participent à toutes les solutions. A cet égard on constate que si les deux algorithmes travaillant sur les coûts réels, à savoir le premier et le troisième, donnent des résultats sensiblement équivalents, la recherche sur les coûts transformés paraît beaucoup plus efficace. En particulier on peut constater que la deuxième méthode dont la procédure de calcul est identique à celle de la première, a toujours donné des résultats meilleurs que cette dernière.

- Il est possible de multiplier les coupures dans l'arbre en initialisant le coût standard à une valeur finie : les coupures sont en effet d'autant plus nombreuses que le coût initial est proche de l'optimal. On est alors certain de n'obtenir que des solutions de coût inférieur à celui-ci. C'est pourquoi dans la pratique les algorithmes de recherche en arbre sont couplés à des heuristiques d'initialisation (par exemple, méthode de KARG pour les problèmes de voyageur).

Temps de calcul.

Il est difficile de caractériser la rapidité des algorithmes proposés dans la mesure où ils n'ont pu aller jusqu'à leur terme. Quoiqu'il en soit, le temps d'une recherche en arbre varie toujours exponentiellement avec le nombre n de variables. Il est de l'ordre de :

- quelques secondes pour n = 10
- quelques minutes pour n = 20

On peut s'attendre à ce que ce temps soit approximativement multiplié par 10 chaque fois que le nombre de variables augmente de 10 unités.

En ce qui concerne les deux premières méthodes testées, les temps annoncés incluent celui qui est nécessaire pour procéder au tri des arcs dans l'ordre de leurs coûts (réels ou transformés) croissants. Le nombre d'arcs à trier est $\frac{n(n-1)}{2}$ et la méthode utilisée est le tri par clés (radix sort). Sa durée est donc proportionnelle à $n^2 \times m$, où m est le nombre maximum de chiffres à trier.

On peut ensuite caractériser le temps d'obtention de la première solution une fois que le tri est réalisé.Dans la plage des valeurs de n considérées, ce temps est proportionnel à n^2. L'évaluation des coefficients de proportionnalité donne :

Temps de tri $\simeq 13.10^{-4}\ n^2$ m
Temps d'obtention de la première solution $\simeq 7.10^{-4}\ n^2$
(algorithmes 1 et 2)

Pour l'algorithme de LITTLE, la première solution est trouvée en un temps proportionnel à n^3 qui peut être approché par :

Temps d'obtention de la première solution $\simeq 4.10^{-4}\ n^3$
(algorithme de LITTLE)

Si on admet que le temps d'obtention de la première solution caractérise la rapidité du parcours de l'arbre, les deux premiers algorithmes l'emportent sur celui de LITTLE.Cependant les bornes calculées par ce dernier sont plus fines ce qui a pour conséquence de multiplier les coupures et donc de diminuer la taille de l'arbre à explorer. Dans ces conditions est-il préférable de parcourir lentement un arbre réduit ou rapidement un arbre important ? Il y a là un compromis difficile à réaliser. Lorsque la matrice des distances est non symétrique, l'algorithme de LITTLE est en général plus performant. Dans le cas contraire, il en va autrement car la symétrie n'apporte aucune simplification notable à cette méthode alors qu'elle permet pour les deux premières de diviser par deux le nombre d'arcs candidats et de simplifier considérablement les critères de compatibilité.

Notons enfin que la rapidité de la recherche peut être améliorée par une transformation simple qui fait intervenir la notion d'écartement : si p_{ij} est le coût de l'arc (i,j), son écartement e_{ij} par rapport au sommet k s'écrit :

$$e_{ij} = p_{ik} + p_{kj} - p_{ij}$$

Le long d'un circuit hamiltonien :

$$\sum e_{ij} = \sum p_{ik} + \sum p_{kj} - \sum p_{ij}$$

Les deux premiers termes du membre de droite sont constants. La recherche du minimum de la somme des p_{ij} est donc équivalente à celle du maximum de la somme des écartements. Cette transformation est intéressante car tous les e_{ik} et e_{kj} sont nuls et le nombre d'arcs à sélectionner n'est plus que n-2. Le gain en temps est de l'ordre de 10 % pour les problèmes traités.

2 - METHODE DES DISTANCES.

La méthode des distances a été utilisée (cf [9] , [16]) pour résoudre les problèmes de voyageur. La différence essentielle avec les méthodes de recherche en arbre réside dans la définition du sous-ensemble des solutions réalisables à explorer dans laquelle intervient une notion de voisinage et donc de distance.

Cependant, aucune étude de convergence n'en a été faite. De plus, cette méthode nous semble être suffisamment générale pour pouvoir être appliquée à une classe plus étendue de problèmes combinatoires.

2.1. - PRINCIPE.

La mise en oeuvre de l'algorithme nécessite qu'on se donne une solution standard initiale x dont le coût est noté C(x).

Nous définissons dans l'espace S des solutions réalisables une distance D qui est une application de S x S dans l'ensemble des entiers de telle sorte qu'elle soit bornée supérieurement. Le choix de cette distance peut varier suivant les problèmes à traiter comme nous le verrons dans les applications. Pour l'instant, on se contente de supposer qu'une telle distance existe.

Une solution réalisable x est alors dite r-optimale s'il n'existe pas x' tel que :

$$C(x') < C(x)$$
$$D(x', x) \leq r$$

Autrement dit, x est la meilleure des solutions contenues dans la boule de centre x et de rayon r notée B(x,r).

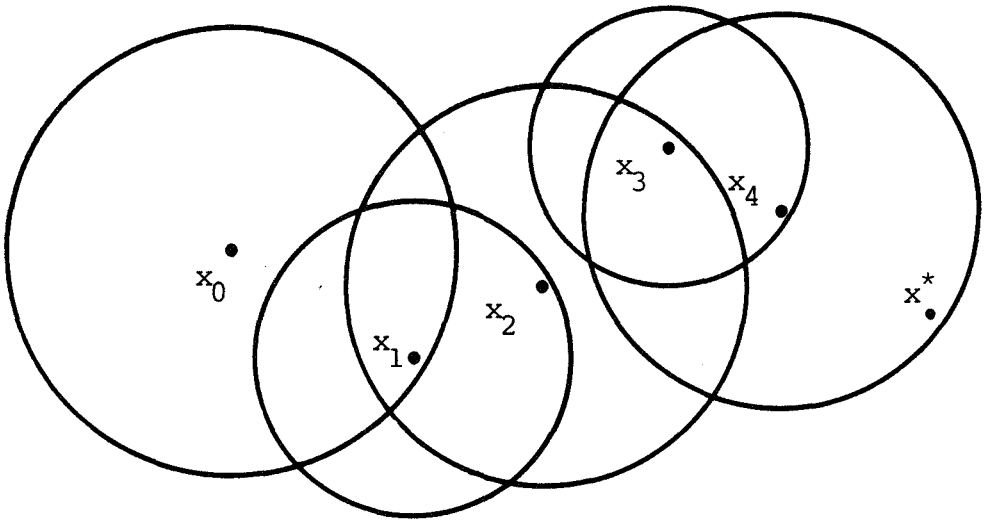
L'algorithme, qui consiste à trouver une suite de solutions réalisables $x_0, x_1, \ldots, x_p$ telle que $C(x_p) < C(x_{p-1}) < \ldots < C(x_0)$ peut s'écrire ainsi :

ETAPE 1 : *Début*
$x \leftarrow x_0$ solution quelconque.

ETAPE 2 : $r \leftarrow 1$

ETAPE 3 : Soit y tel que C(y) = minimum C(t), $t \in B(x,r)$
Si $y \neq x$ *alors* $y \leftarrow x$ *aller à* Etape 2,
sinon si $r < R$ *alors* $r \leftarrow r+1$ *aller à* Etape 3,
sinon fin.

Partant d'une solution quelconque x_0, la méthode procède à une énumération des solutions réalisables dans une boule centrée sur x_0 et de rayon r. La meilleure de ces solutions, si son coût est inférieur à celui de x_0, est retenue pour être le centre d'une nouvelle boule d'exploration.



Si on n'imposait pas de limite R à r, on serait conduit à énumérer toutes les solutions réalisables du problème. En effet, si r devient suffisamment grand, la solution optimale x^* se trouve à coup sûr dans une boule de rayon r, quel que soit son centre. Ceci résulte du fait que la distance donnée est bornée supérieurement sur N. Dans la mesure où r a une limite R, la solution trouvée n'est que R-optimale. Le problème est de savoir dans quelle condition elle constitue l'optimum absolu.

2.2. CONVERGENCE.

Supposons que nous soyons capables de trouver une distance sur S^2 qui, outre les propriétés déjà énoncées, soit telle qu'il existe une régression linéaire significative entre :

$$d(x) = D(x, x^*) \text{ et } c(x) = C(x) - C(x^*)$$

On peut alors énoncer le théorème :

THEOREME.

Si x_k est le $k^{ième}$ itéré de l'algorithme situé à une distance h de x^*, alors il existe $\mu>0$, constante indépendante de k, telle que la probabilité de stationner en x_k soit inférieure à :

$$\prod_{t=1}^{R} \left(\frac{\mu}{t^2}\right)^{\text{card}(A_t)} \qquad \text{où} \quad A_t = B\,(x_k,R) \cap B\,(x^*,h-t)$$

PREUVE.

La régression supposée permet d'écrire :

$$c(x) = \alpha\, d(x) + z$$

où z est une variable aléatoire de moyenne $\bar{c} - \alpha\bar{d}$.

On peut également écrire :

$$c(x) = \bar{c} + \alpha\,(d(x) - \bar{d}) + \varepsilon$$

où ε est le résidu linéaire non corrélé, dont l'espérance est nulle et dont la variance est donnée par :

$$\sigma^2(\varepsilon) = \sigma^2(z) = (1-\rho^2)\,\sigma^2(c)$$

si ρ désigne le coefficient de corrélation.

On a également :

$$\alpha = \rho \frac{\sigma(c)}{\sigma(d)}$$

Le coût de la solution x_k, située à une distance h de x^*, peut se mettre sous la forme :

$$C(x_k) = \alpha h + z_0$$

Parmi les solutions explorées dans $B(x_k,R)$, il s'en trouve un certain nombre dont les distances à x^* sont h-1, h-2,..., h-t,..., h-R.

Soit y_i^t une solution située à une distance h-t de x^*. Son coût s'écrit :

$$C(y_i^t) = \alpha(h-t) + z_i$$

x_k est stationnaire si $C(y_i^t) > C(x_k)$ quels que soient i et t. Or :

$$C(y_i^t) > C(x_k) \longleftrightarrow z_0 - z_i > \alpha t$$

L'inégalité de TCHEBICHEFF donne :

$$P\{|z_0-z_i| > \alpha t\} < \frac{\sigma^2(z_0-z_i)}{\alpha^2 t^2} = \frac{2(1-\rho^2)}{\rho^2}\,\frac{\sigma^2(d)}{t^2}$$

Si on pose $\mu = 2\,\frac{1-\rho^2}{\rho^2}\,\sigma^2(d)$, cette expression devient :

$$P\{|z_0-z_i| > \alpha t\} < \frac{\mu}{t^2}$$

La stationnarité n'est réalisée que si toutes les solutions y_i^t sont moins bonnes que x_k. La probabilité cherchée est donc bornée par :

$$\prod_{t=1}^{R} \left(\frac{\mu}{t^2}\right)^{card(A_t)}$$

où

$$A_t = B(x_k, R) \cap B(x^*, h-t)$$

Remarque :

Cette limite est très grossière. Elle n'offre que peu d'intérêt pratique dans la mesure où même si la corrélation est forte, elle est supérieure à 1. Si on désire calculer une probabilité de stationnarité plus fine, il faut évaluer $P\{ z_0 - z_i > \alpha t\}$ autrement que par l'inégalité de TCHEBICHEFF, en construisant la loi de répartition de $z_0 - z_i$. Pour des valeurs du coefficient de corrélation proches de 1, il paraît tout à fait justifié d'assimiler z à une variable aléatoire gaussienne. Dans ces conditions, si ω désigne une variable aléatoire gaussienne centrée réduite :

$$P\{z_0 - z_i > \alpha\, t\} = P\left\{\omega > \frac{\rho}{2(1-\rho^2)} \quad \frac{t}{\sigma(d)}\right\}$$

Une autre difficulté vient du fait qu'il est souvent impossible d'évaluer correctement card (A_t) ; tout au plus peut on en trouver une borne inférieure.

Le résultat important est que la probabilité de convergence vers l'optimum, c'est-à-dire de non stationnarité, est d'autant plus élevée que la corrélation est plus forte et que la variance de la distance est plus faible.

2.3. APPLICATION AUX PROBLEMES D'ORDONNANCEMENT.

L'utilisation de la méthode précédente implique le choix d'une distance bornée sur l'espace des solutions réalisables et dont la corrélation avec les coûts soit significative. Ce choix dépend de la nature du problème à traiter. Ceux que nous avons abordés sont de type n|1 pour lesquels une solution réalisable est notée :

$$x = (i_1 - i_2 - i_3 - \ldots - i_n)$$

Lorsque le critère fait intervenir la notion de retard, une bonne distance entre deux solutions consiste à prendre le nombre minimal de permutations d'éléments voisins $i_k - i_{k+1}$ qui permet de passer de l'une à l'autre. Pour le problème de voyageur, nous avons choisi le nombre d'arcs non communs.

Exemple

Si n = 8 et si x, y et z représentent les circuits suivants :

$$x = (1 - 2 - 3 - 4 - 5 - 6 - 7 - 8)$$
$$y = (1 - 3 - 2 - 6 - 8 - 7 - 5 - 4)$$
$$z = (1 - 3 - 2 - 6 - 7 - 8 - 5 - 4)$$

Les distances qui les séparent, en supposant la matrice symétrique, sont données par :

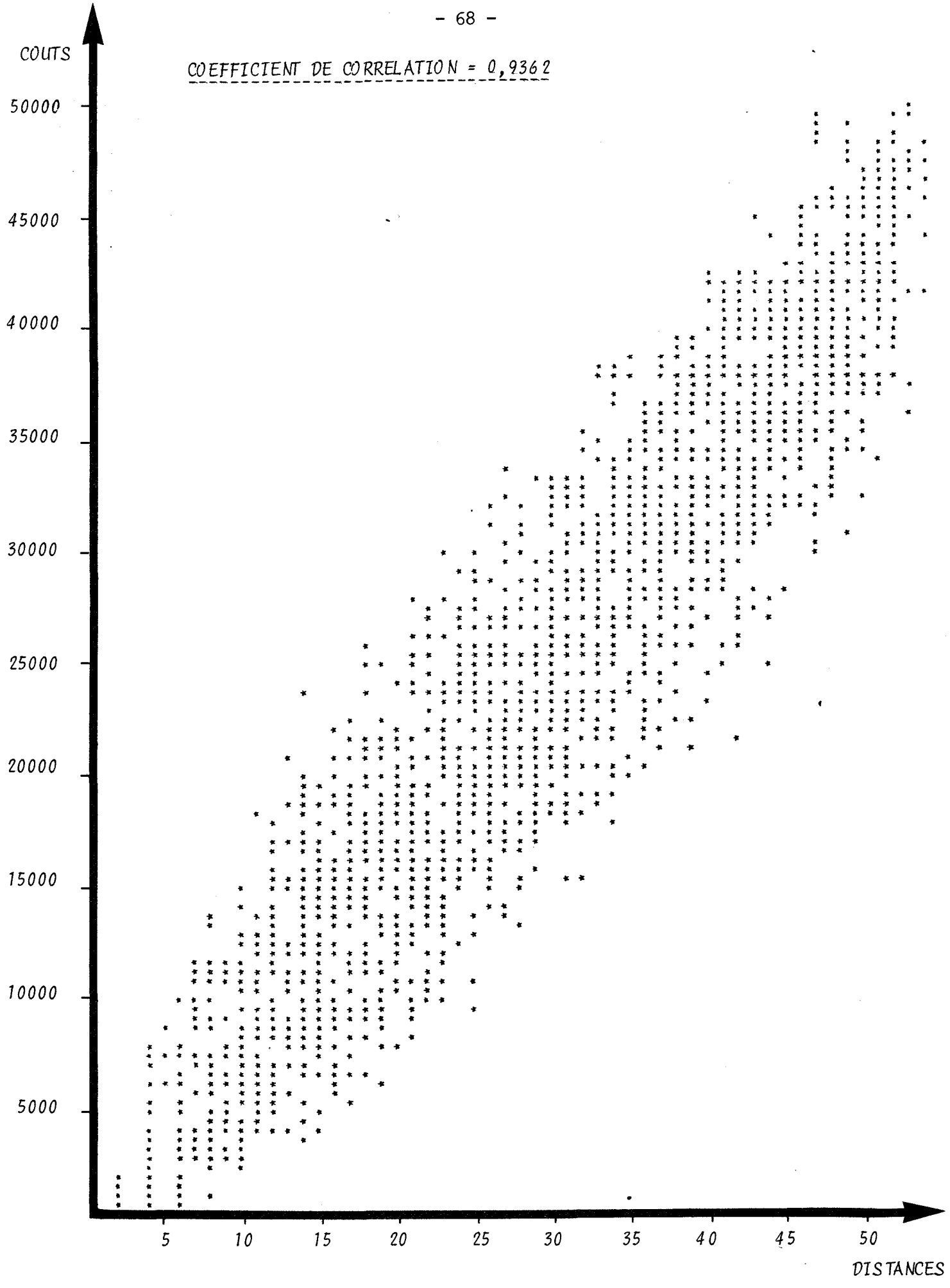
$$D(x, y) = 5$$
$$D(x, z) = 4$$
$$D(y, z) = 2$$

On vérifie facilement que la distance ainsi définie possède les propriétés :

$$1 - D(x,x) = 0$$
$$2 - D(x,y) = D(y,x)$$
$$3 - D(x,y) + D(y,z) \geq D(x,z)$$

En outre, elle est bornée supérieurement sur l'espace des solutions par le nombre d'arcs du circuit, c'est-à-dire par le nombre n de villes. Il faut enfin vérifier que la corrélation coût-distance est significative. Le coefficient a été calculé pour les trois problèmes testés sur un échantillon de 5000 solutions réalisables. Sa valeur a toujours été supérieure à 0,9. Le graphique ci-après donne l'allure du nuage de points représentatifs de l'échantillon rapporté à un plan coût-distance pour n=57.

Lorsque la matrice est symétrique, la plus petite distance entre deux solutions différentes est égale à 2. Le nombre de solutions ayant exactement deux arcs non communs avec le centre de la boule d'exploration est $\frac{n(n-3)}{2}$. Le nombre de celles qui en ont trois est $\frac{n(n-4)(2n-7)}{3}$. Elles sont générées par inversion et insertion.


COUTS
COEFFICIENT DE CORRELATION = 0,9362
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
5
10
15
20
25
30
35
40
45
50
DISTANCES

Si $(i_1 - i_2 -\dots - i_{j-1} - \overline{\left| i_j - i_{j+1} \dots i_{k-1} - i_k \right|} - i_{k+1} - \dots - \ln)$

désigne le centre de la boule, les solutions qui en sont à une distance égale à 2 sont de la forme :

$$(i_1 - i_2 -\dots - i_{j-1} - \overline{\left| i_k - i_{k-1} -\dots - i_{j+1} - i_j \right|} - i_{k+1} -\dots - i_n$$

elles sont obtenues par inversion de la chaîne $i_j -\dots- i_k$. Une solution 2-optimale est donc stable par rapport à l'inversion.

Les solutions distantes de 3 du centre sont données par la formule générale:

$$(\overline{\left| i_j - i_{j+1} -\dots- i_{k-1} - i_k \right|} - i_1 - i_2 -\dots- i_{j-1} - i_{k+1} -\dots- i_n)$$

Les trois solutions générées à partir de cette dernière par inversion des chaînes $(i_j -\dots- i_k)$, $(i_1 -\dots- i_{j-1})$, $(i_{k+1} -\dots- i_n)$ appartiennent également à la boule de rayon 3. Une solution 3-optimale est donc stable par rapport à l'insertion et l'inversion.

Compte tenu du temps nécessaire à la génération de ces solutions,qui est en première approximation proportionnel à leur nombre, nous n'avons pas envisagé d'explorer des boules dont le rayon soit supérieur à 3.

2.4. - EVALUATION DES PERFORMANCES.

L'algorithme qui a été programmé diffère légèrement de celui qui a été présenté au paragraphe 2.1, en ce sens que le recentrage de la boule s'effectue non pas sur la meilleure des solutions qui y sont incluses mais sur la première solution détectée de coût inférieur à celui du centre. En outre, on a fait en sorte que les insertions ou les inversions qui ne conduisaient pas à une diminution du coût ne soient pas générées plusieurs fois de suite.

Les performances de l'algorithme ainsi défini ont été, dans un premier temps, testées sur des matrices aléatoires symétriques, dont les ordres sont compris entre 10 et 30. Pour chacune d'elles, on a effectué 100 essais à partir d'initialisations aléatoirement indépendantes avec R=2 puis R=3, afin d'évaluer le temps d'exploration des boules et une probabilité empirique de convergence vers l'optimum . Si l'on désigne par p_2 la probabilité pour qu'une solution 2-optimale soit l'optimum absolu, par T_2 le temps en secondes nécessaire pour trouver cette solution, par p_3 et T_3 les mêmes grandeurs lorsque R=3, les essais ont donné les résultats suivants :

Qualité de la solution.

p_i \ n	10	15	20	25	30
p_2	0,32	0,24	0,10	0,02	0,02
p_3	1,00	0,86	0,37	0,38	0,30

Les probabilités de convergence vers la solution optimale peuvent être approchées par les formules :

$$p_2 \simeq \frac{400}{n^3}$$

$$p_3 \simeq \frac{10}{n}$$

Temps de calcul.

En plus des essais précédents pour lesquels ont été calculées les moyennes des temps de calcul, on a appliqué une fois l'algorithme sur des matrices dont le rang est compris entre 35 et 55.

T_i \ n	10	15	20	25	30	35	40	45	50	55
T_2	0,15	0,40	0,84	1,24	2,11	2,45	3,77	4,89	4,85	10,10
T_3	0,51	3,08	8,73	10,67	39,64	60,56	98,91	180,15	247,87	261,04

on a alors :

$$T_2 \simeq 2{,}2\ n^2 .10^{-3}$$

$$T_3 \simeq 4{,}5\ n^{10/3} .10^{-4}$$

Les temps de mise en oeuvre de l'algorithme étant assez courts, on peut se permettre de l'itérer un certain nombre de fois avec des solutions initiales différentes. Si k est le nombre d'itérations, la probabilité de trouver la solution optimale devient :

$$p = 1 - (1 - p_R)^k$$

En un temps de calcul

$$T = k\ T_R$$

p_R et T_R désignent respectivement la probabilité de succès et le temps de calcul associés à la distance maximale R.

Dans ces conditions, pour une probabilité de succès désirée, les calculs sont plus rapides avec R = 3.

Une série de tests a été réalisée sur les problèmes de voyageur à 33 et 42 variables afin de vérifier l'exactitude des calculs précédents. Pour chacun d'eux, on indique le nombre n de variables du problème, le rayon maximal R d'exploration, le nombre k d'itérations, la probabilité de succès p à laquelle on peut s'attendre, et le temps de calcul T, ces deux dernières grandeurs étant évaluées à l'aide des formules que nous avons établies :

n	R	k	p	T
33	2	100	0,68	240
33	3	6	0,98	310
42	2	50	0,24	195
42	3	3	0,56	350

Pour chacun de ces quatre problèmes, on a effectué 10 essais afin de comparer les fréquences de succès observées f aux probabilités qui ont été calculées. On a également mentionné les valeurs minimales et maximales des coûts de ces 10 solutions.

n	R	MINIMUM	MAXIMUM	δ
33	2	10861	10925	0,9
33	3	10861	10861	1,0
42	2	699	705	0,4
42	3	699	713	0,5

Les valeurs de f sont donc très proches de celles de p. En outre, les coûts des solutions les plus mauvaises sont voisins du coût optimal, ce qui signifie que même si l'algorithme n'est itéré qu'une fois, on est certain d'obtenir un résultat très satisfaisant.

Une variante de la méthode consiste à fournir comme centre de la boule initiale non plus une solution aléatoire, mais celle qui est fournie par l'heuristique de KARG. Aucune modification sensible des performances n'a été constatée, même en ce qui concerne les temps de calcul : à partir d'une "mauvaise" initialisation, l'algorithme des distances fournit extrèmement rapidement une solution comparable à celle qui est produite par cette heuristique. En effet, la probabilité de stationnarité varie exponentiellement avec card (A_t) dont la valeur est grande lorsque la solution initiale est éloignée de l'optimale : la moindre inversion ou insertion permet de s'en approcher et donc d'améliorer le coût. Au fur et à mesure qu'on tend vers l'optimum, card (A_t) diminue, la rapidité de la convergence également, jusqu'à ce qu'elle s'annule lorsqu'on atteint la solution R-optimale. On peut enfin remarquer que les performances sont très supérieures à celles des méthodes de recherche en arbre ; ceci s'explique par le fait qu'un échantillon beaucoup plus varié de solutions réalisables a été parcouru : alors que dans un cas ces solutions sont très semblables puisque seuls les derniers arcs retenus diffèrent, la notion de voisinage permet d'effectuer des substitutions à tous les niveaux.

3 - INTERET DE CES METHODES.

L'intérêt essentiel de ces méthodes, par rapport aux heuristiques non convergentes, réside dans la qualité de la solution fournie. Initialisées aléatoirement, elles sont en général plus performantes qu'une heuristique très appropriée, et n'ont en tous cas jamais fourni de mauvais résultats. La possibilité de les initialiser en les couplant à cette heuristique assure qu'elles sont toujours au moins aussi efficaces.

Elles présentent également l'intérêt d'être beaucoup plus générales dans leur principe : les divers algorithmes de recherche en arbre ne diffèrent essentiellement que par le mode de calcul des bornes, les méthodes des distances que par la définition d'une distance adaptée au problème à résoudre.

Leur seul défaut pourrait résider dans la longueur du temps nécessaire à leur mise en oeuvre. En fait, cet inconvénient n'existe pas : la première solution standard est en effet atteinte très rapidement, toujours au moins aussi vite que par une heuristique non convergente puisqu'elle peut être fournie par cette dernière ; l'algorithme peut être arrêté arbitrairement quand on le désire, la solution standard du moment constituant la réponse cherchée. De ce fait, la longueur de la procédure est plutôt un avantage puisqu'elle permet d'exploiter pleinement le temps de calcul alloué.

CHAPITRE 4

LES HEURISTIQUES D'APPRENTISSAGE

C H A P I T R E IV

LES HEURISTIQUES D'APPRENTISSAGE

Le terme d'apprentissage englobe toutes les méthodes dans lesquelles les résultats d'expériences passées sont utilisés pour élaborer une solution du problème.

Lorsqu'il est confronté à une situation totalement inconnue, l'être humain détermine son comportement au hasard, et il enregistre les conséquences de ses décisions dans sa mémoire. Lorsqu'un problème identique ou similaire se présente par la suite, il cherchera à évaluer la portée d'une autre décision possible. Si celle-ci lui apporte des résultats plus satisfaisants que la première qu'il avait envisagée, il aura naturellement tendance à la lui préférer chaque fois que l'occasion s'en présentera dans l'avenir. C'est ainsi qu'il acquiert petit à petit une "expérience" qui lui dicte instinctivement la ou les conduites à tenir en face de chaque situation. Cette expérience n'est pas figée ; elle peut toujours être enrichie et complétée par l'étude des décisions qui avaient été écartées jusque là.

L'apport essentiel de l'informatique dans ce domaine réside en ce que les décisions peuvent être simulées, et donc que l'expérience peut être acquise très rapidement. Les premiers chercheurs à s'être intéressés à ce problème sont ceux qui ont voulu apprendre à l'ordinateur à jouer aux dominos, dames, etc... SAMUEL [14] en particulier a tenté d'implémenter sur machine le jeu d'échec.

Une méthode de résolution classique, lorsque l'ordinateur doit décider d'un coup à jouer, pour une disposition des pièces sur l'échiquier donnée, consiste à envisager toutes les stratégies possibles et à choisir la meilleure d'entre elles. A cette fin, il effectue une recherche en arbre.

- les niveaux impairs concernent les coups jouables par l'ordinateur.
- les niveaux pairs concernent toutes les ripostes possibles de l'adversaire.

En principe, il faudrait développer l'arborescence suffisamment profondément pour aboutir à des feuilles terminales, c'est-à-dire des situations où la partie est nulle, gagnée ou perdue pour l'un des deux partenaires. Cette recherche est bien évidemment impossible à réaliser complètement car le problème devient rapidement très combinatoire : même si le nombre de pièces est assez réduit, chacune d'ellespossède en général un nombre de degrés de liberté important, ce qui multiplie à l'infini les possibilités. Elle est d'autant plus impraticable que le règlement du jeu prévoit un délai de réflexion limité avant chaque coup. La solution qui a été adoptée consiste à réduire la taille de l'arbre de recherche en ne poussant pas l'exploration jusqu'aux feuilles terminales. Dans un premier temps, on a détecté par apprentissage des situations-types relativement simples qui aboutissent à terme au succès de l'un ou l'autre des joueurs. Par la suite, ces mêmes méthodes ont permis d'affecter à des situations plus complexes une "note", plus ou moins bonne suivant qu'elles sont par expérience favorables ou non, qui dépend des valeurs prises par un certain nombre de paramètres caractéristiques.

Les heuristiques d'apprentissage diffèrent les unes des autres par la notion de situation, et par la manière dont est évaluée la portée d'une décision. La principale difficulté réside en ce qu'il est en général peu commode de caractériser les situations et donc de définir leur similarité ou leur dissemblance. Le plus souvent, ce problème est résolu par la construction d'une fonction d'une ou plusieurs variables qui permet de les regrouper en classes d'équivalence. Le tout est de savoir quelles sont les variables influentes ; ce ne sont pas toujours les plus immédiates.

Dans le domaine de l'ordonnancement, nous distinguerons deux types d'apprentissage :

1 - *L'apprentissage des méthodes* qui sera utilisé pour résoudre le problème d'atelier, dérive directement du précédent. Il consiste à apprendre quelle décision adopter face à une situation donnée. Pour effectuer ce choix, on dispose d'un certain nombre de méthodes heuristiques parmi lesquelles il faudra en sélectionner une en chaque point de décision. Une solution réalisable est donc déterminée par une certaine combinaison de ces heuristiques. Le problème est de trouver quelle est celle qui conduit au meilleur résultat.

2 - *L'apprentissage des caractéristiques* diffère de celui des méthodes, en ce sens que les solutions réalisables sont construites à l'aide d'une seule heuristique. Nous avons vu dans les chapitres antérieurs que chaque fois que cela était possible, on itérait une méthode de résolution pour conserver la meilleure des solutions de l'espace parcouru. Or on constate que très souvent ces solutions, sans être absolument équivalentes, possèdent des caractéristiques communes. On fait alors l'hypothèse que toute bonne solution doit les posséder. Il faut donc les extraire de l'échantillon construit ; c'est ce qui est réalisé par ce type d'apprentissage qui sera mis en oeuvre pour résoudre les problèmes de voyageur.

1 - APPRENTISSAGE DES METHODES.

1.1. - PRINCIPE

La construction d'une solution réalisable d'un problème est équivalente à la prise d'un nombre fini de décisions. Le problème est combinatoire lorsque les décisions possibles sont nombreuses. Il faut donc, à chaque fois que l'occasion s'en présente, effectuer un choix. Une manière simple de le réaliser consiste à utiliser systématiquement la même heuristique :

Exemple :

- Dans un premier temps on pourra se contenter de prendre la décision la moins immédiatement coûteuse, la plus rentable à brève échéance.

- Si on travaille à plus long terme, d'autres critères de choix sont possibles, qui peuvent être contradictoires avec les précédents.

La solution ainsi construite est d'autant meilleure que l'heuristique est bien appropriée au problème traité. Or il est souvent impossible d'affirmer à priori que telle ou telle méthode sera plus performante que les autres. Il arrive même que certaines d'entre elles, qui s'avèrent d'une manière générale efficaces, fournissent dans des cas particuliers des résultats décevants (voir à ce sujet l'étude des heuristiques non convergentes appliquées au problème d'atelier, Chapitre 2 § 13). On observe également que des critères de choix qui semblent au premier abord opposés, produisent des résultats de qualité comparable : ceci s'explique par le fait que chacun d'entre eux n'est adapté qu'à un nombre limité de situations. Leur utilisation systématique, c'est-à-dire indépendamment de la situation devant laquelle on se trouve, fait qu'un certain nombre de mauvaises décisions ont été prises.

Toutes ces raisons conduisent à penser qu'il serait préférable d'utiliser non plus une seule, mais plusieurs heuristiques que l'on combinerait entre elles pour définir un ensemble de décisions cohérentes plus appropriées aux diverses situations qui peuvent se présenter.

PROBLEME.

Donnons-nous donc n heuristiques H_1, H_2,..., H_n et supposons que les situations possibles sont au nombre de m ; nous ne chercherons pas à en donner une caractérisation précise car celle-ci dépend essentiellement de la nature du problème. Il s'agit de trouver l'heuristique H_i, $i \leq i \leq n$ la plus efficace dans la situation j, $1 \leq j \leq m$.

METHODE.

Cette recherche est effectuée par apprentissage : on associe à chacune de ces situations un vecteur probabilité p_j à n composantes ; $p_j(i)$ est la probabilité que l'heuristique H_i soit la plus adaptée à la situation j.

Initialement, nous ne savons pas laquelle est la meilleure ; nous aurons donc :

$$p_j(i) = \frac{1}{n} \qquad 1 \leq i \leq n \quad 1 \leq j \leq n$$

Le problème sera résolu lorsque pour tout j :

$$p_j(k) = 1$$

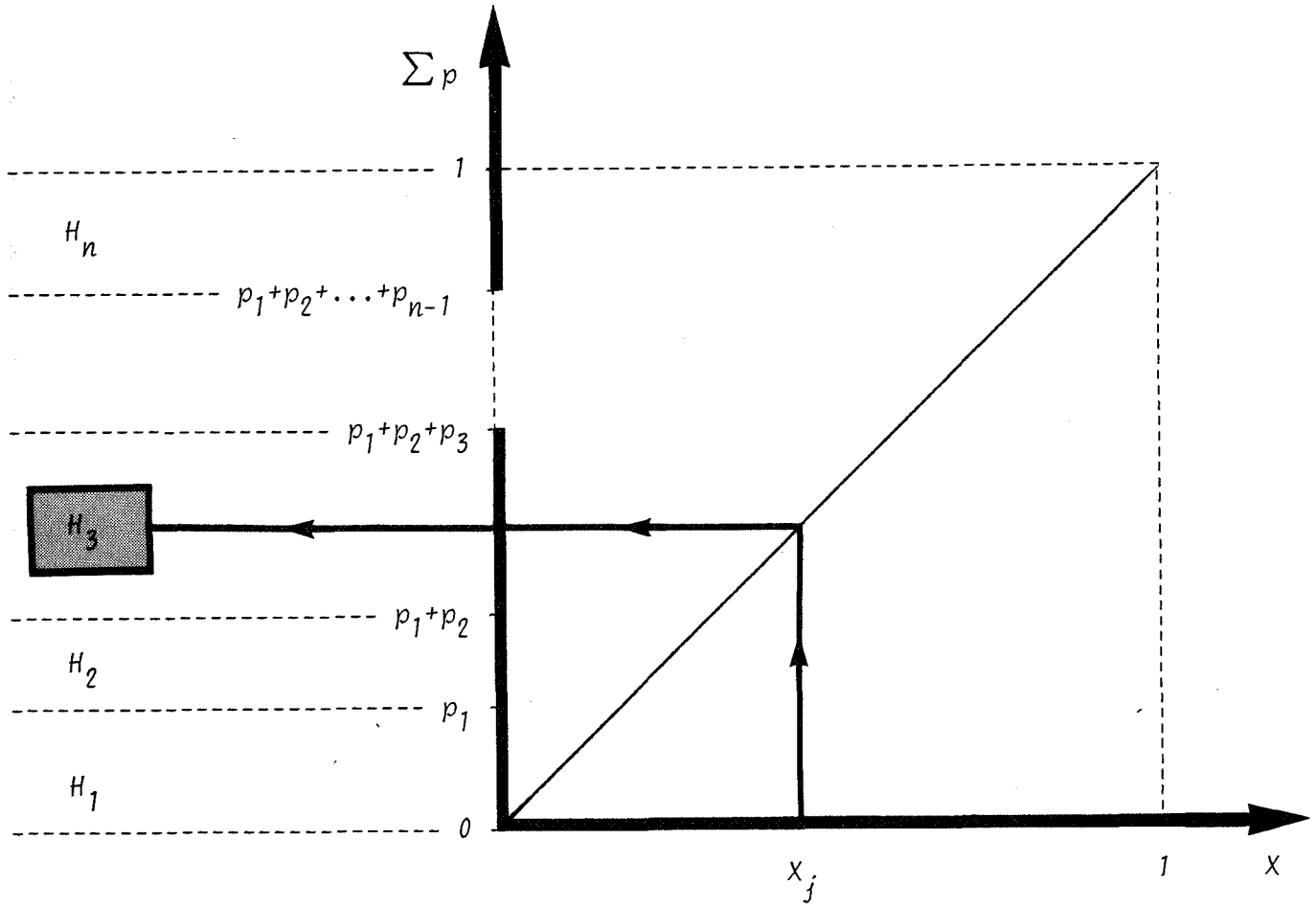
$$p_j(i) = 0 \qquad 1 \leq i \leq n \qquad i \neq k$$

H_k est alors l'heuristique cherchée.

Avant de mettre en route la procédure qui permet de faire converger le système de probabilités ainsi défini, il faut encore se donner une solution standard initiale. Ce pourra être par exemple la meilleure de celles auxquelles conduit l'application systématique de chacune des n heuristiques. L'acquisition de l'expérience peut alors être simulée: on procède au tirage de m nombres aléatoires X_j compris entre 0 et 1. L'heuristique sélectionnée dans la situation j est H_i si :

$$\sum_{k=1}^{i-1} p_j(k) \leq X_j < \sum_{k=1}^{i} p_j(k)$$

Cette règle peut être interprétée graphiquement :



Choix de l'heuristique en fonction de X_j

La connaissance des règles de prise de décision permet de construire la solution réalisable correspondante . Son coût est comparé à celui de la solution standard ; deux cas peuvent se produire :

- S'il est de meilleure qualité, la dernière solution trouvée devient la nouvelle solution standard. On admet que toutes les décisions prises ont eu des conséquences favorables : pour en tenir compte dans l'avenir, on augmente la probabilité des heuristiques choisies d'un incrément e ($0 \leq e \leq 1$), et on diminue celle des autres de $\frac{e}{n-1}$.

- Si elle est moins bonne, on augmente de e les probabilités de choix des règles qui ont contribué à la construction de l'ordonnancement standard, et on diminue les autres de $\frac{e}{n-1}$.

L'algorithme peut donc être résumé de la manière suivante :

ETAPE 1 : *Début*

$p_j(i) \leftarrow \frac{1}{n}$ $\quad 1 \leq i \leq n$

$\quad 1 \leq j \leq m$

Calculer le coût C de la solution standard initiale.

ETAPE 2 : Choisir aléatoirement une heuristique de décision pour chaque situation.

Calculer le coût C' de la solution à laquelle conduit l'utilisation de ces heuristiques.

Si $C' < C$ *alors* $C \leftarrow C'$

solution standard ← dernière solution trouvée.

Incrémenter de e la probabilité de choix des heuristiques ayant fourni la solution standard,

décrémenter de $\frac{e}{n-1}$ la probabilité de choix des autres heuristiques.

Si $\exists$ k tel que $p_j(k) = 1$ $\quad 1 \leq j \leq m$

alors fin

sinon aller à Etape 2.

Algorithme d'apprentissage des solutions

1.2. - ETUDE DE LA CONVERGENCE.

L'algorithme présenté est convergent. En effet, le coût des solutions est borné par le coût optimal et la suite des coûts standards est monotone. Ceci a pour conséquence qu'au bout d'un nombre fini de simulations cette suite devient stationnaire. En conséquence, les probabilités associées aux heuristiques qui ont déterminé la construction du dernier standard trouvé tendent vers 1.

La qualité des solutions apportées par cet algorithme et la rapidité de la convergence dépendent essentiellement de plusieurs facteurs :

1 - Heuristiques de choix proposées : elles sont en général en nombre limité, ceci pour des contraintes de temps de calcul, et leur combinaison ne permet que la génération d'un sous-ensemble limité de solutions réalisables. On a tout intérêt à n'en choisir que deux ou trois au maximum parmi celles dont l'application systématique en chaque point de décision conduit aux meilleurs résultats.

2 - Caractérisation des situations : si elle n'est pas correctement réalisée, on risque de trouver dans une même classe d'équivalence des situations différentes pour lesquelles les heuristiques les plus adaptées ne sont pas nécessairement identiques. A la limite, cela veut dire qu'une décision prise aléatoirement a toutes chances de donner de bonsrésultats puisqu'elle est la meilleure pour au moins un élément de la classe. Dans ces conditions, l'heuristique vers laquelle on converge est celle qui a été la plus favorisée par le tirage aléatoire, alors qu'en principe elle ne devait pas dépendre de ce tirage.

3 - Valeur de l'incrément : nous avons choisi de donner à e une valeur constante. Plus cette valeur est grande, plus la convergence est rapide, mais alors le sous-espace exploré est réduit et on a moins de chances d'améliorer la solution standard. Il y a là encore un compromis qualité-temps de calcul à réaliser. De plus,rien n'interdit de faire varier cette valeur avec le temps. Au cours des premières simulations, elle sera faible afin de permettre l'exploration du voisinage du standard initial le plus vaste possible. Par la suite, on l'augmentera progressivement afin d'assurer une convergence rapide.

1.3. - APPLICATION AUX PROBLEMES D'ORDONNANCEMENT [11]

La méthode d'apprentissage telle que nous l'avons décrite a été testée sur les problèmes d'atelier. Les trois points que nous venons de soulever ont été résolus de la manière suivante :

1 - Heuristique de choix : l'étude des méthodes de résolution non convergentes des problèmes d'atelier a montré que l'on pouvait faire une distinction entre deux catégories d'heuristiques : d'une part celles qui favorisent les tâches courtes, d'autre part, celles qui privilégient les pièces dont les durées d'exécution sont les plus longues. On a pu également constater que suivant les problèmes, la meilleure solution était détectée en appliquant tantôt les unes, tantôt les autres. Ceci peut s'expliquer par le fait qu'au début du processus d'ordonnancement, on a en général intérêt à faire passer les pièces qui permettent d'alimenter au plus vite l'ensemble des machines ; par la suite, ce sont celles qui nécessitent le plus d'heures de travail qui doivent être choisies en priorité, afin de ne pas retarder leurs dates d'achèvement. Il paraît donc naturel de ne prendre en compte que deux critères de choix appartenant à l'une et à l'autre de ces catégories. Dans la première d'entre elles, deux méthodes ont fourni des résultats comparables, ce sont CD et PR-D/TR. Dans la seconde, c'est LTR qui s'est montrée la plus efficace. Nous avons donc essayé deux apprentissages à deux heuristiques de choix, l'une faisant intervenir CD et LTR, l'autre PR-D/TR et LTR.

2 - Caractérisation des situations : les situations de prise de décision sont celles où une pièce vient d'être terminée sur une machine et où il faut donc de nouveau l'alimenter. Les paramètres qui les caractérisent doivent donc tenir compte de la machine et des en-cours ; Ce pourraient être par exemple :

- La position de la machine (se trouve-t-elle toujours en début ou en fin de gamme ?)
- Son degré de saturation (peut-on se permettre des temps morts ?).
- Le nombre et la nature des pièces en attente.
- L'état d'avancement de l'ensemble des travaux.
- etc...

L'extraction des plus influents d'entre eux est un problème délicat ; nous nous sommes contentés d'une approche simple qui consiste à définir une situation par le numéro de la machine et par le rang de la décision qui doit être prise. Le nombre des situations différentes est donc égal à celui des points de décision, c'est-à-dire à celui des tâches à effectuer.

3 - Valeur de l'incrément : elle est constante et égale à 0,0025 ce qui assure la convergence en un temps maximal de l'ordre de 10 minutes. Les simulations sont arrêtées, lorsque pour chaque situation l'une des probabilité est au moins égale à 0,9. Il est en effet inutile de les poursuivre puisqu'alors les coûts obtenus sont pratiquement tous égaux au dernier coût standard.

1.4. - EVALUATION DES PERFORMANCES.

L'apprentissage 1 est réalisé sur les heuristiques CD et LTR, l'apprentissage 2 sur PR-D/TR et LTR. Chaque colonne des tableaux de résultats est relative à une amélioration de la solution standard qui est caractérisée par son coût, le numéro de la simulation qui l'a produite, et le temps en secondes qui s'est écoulé depuis le début de la mise en oeuvre de l'algorithme. La première de ces colonnes concerne le standard initial qui est construit par application systématique de la meilleure des deux heuristiques de la méthode en chaque point de décision, et la dernière définit le point de convergence.

Méthode \ Problème		6 x 6			
APPRENTISSAGE 1	COUT STANDARD	61	60	59	59
	NUMERO SIMULATION	0	5	8	174
	TEMPS	0,6	2,1	3,0	52,8
APPRENTISSAGE 2	COUT STANDARD	61	60	59	59
	NUMERO SIMULATION	0	2	4	166
	TEMPS	0,6	1,2	1,8	50,4

Problème / Méthode	10 x 10							
APPRENTISSAGE 1	COUT STANDARD	1074	1055	1042	1030	1029	994	994
	NUMERO SIMULATION	0	11	24	37	170	181	520
	TEMPS	2,2	14,3	28,6	42,9	189,2	201,3	574,2
APPRENTISSAGE 2	COUT STANDARD	1017	1015	1013	984	984		
	NUMERO SIMULATION	0	48	104	106	370		
	TEMPS	2,2	55,0	116,6	118,8	409,2		

Problème / Méthode	20 x 5		
APPRENTISSAGE 1	COUT STANDARD	1267	1248	1248
	NUMERO SIMULATION	0	136	430
	TEMPS	2,8	193,2	604,8
APPRENTISSAGE 2	COUT STANDARD	1259	1257	1257
	NUMERO SIMULATION	0	128	414
	TEMPS	2,8	180,0	582,4

Qualité de la solution.

Dans tous les cas, le standard initial a été amélioré. Les coûts auxquels on a abouti sont tous inférieurs aux meilleurs de ceux qui avaient été fournis par les heuristiques non convergentes. On peut également constater que pour un même problème, les solutions vers lesquelles on converge ne semblent pas dépendre de l'apprentissage utilisé : les coûts sont très voisins et les temps de calcul comparables.

Il est bien certain que ces résultats auraient pu être améliorés par une mise en oeuvre plus fine de la procédure, notamment en ce qui concerne la définition des situations et le choix des valeurs de e, ainsi que par une étude plus rigoureuse du meilleur système de probabilités initial. Nous avons simplement voulu montrer comment l'apprentissage permet, à partir d'heuristiques non convergentes, de définir une méthode de résolution convergente qui ne peut qu'en améliorer les performances.

Temps de calcul

Les seuls temps auxquels on ait aisément accès sont ceux qui sont nécessaires à la réalisation d'une simulation. Ce sont respectivement 0,3 s - 1,1 s - 1,4 s pour les trois problèmes traités; il est à noter qu'ils sont légèrement inférieurs à ceux qui ont été produits au chapitre 2, car les durées d'entrée-sortie et d'initialisation sont maintenant réparties sur l'ensemble des simulations. Quoiqu'il en soit, ils peuvent être approchés par les mêmes expressions, au coefficient de proportionnalité près.

La convergence est atteinte en un temps qu'il est impossible en pratique de définir à l'aide d'une formule. La seule chose que l'on puisse affirmer est qu'il est d'autant plus court que la valeur de e est grande. De toutes les manières, la procédure peut être arrêtée quand on le désire, ce qui rend l'utilisation de telles méthodes toujours possible dans un contexte temps réel.

2 - APPRENTISSAGE DES CARACTERISTIQUES.

2.1. - PRINCIPE.

Comme nous l'avons déjà indiqué, cette forme d'apprentissage est employée lorsqu'un algorithme peut être itéré un certain nombre de fois pour construire un échantillon de solutions réalisables ; le processus de résolution que nous avons utilisé jusqu'ici consiste à comparer chacun des éléments de cet échantillon, au fur et à mesure qu'ils sont générés, à la solution standard du moment ;

celle-ci est alors éventuellement mise à jour. Autrement dit, nous ne nous intéressions qu'à leurs coûts et absolument pas à leurs structures.

Or, il arrive fréquemment que ces structures présentent des similitudes, en général d'autant plus accentuées que les solutions sont de bonne qualité et donc proches de l'optimum. Si ces caractéristiques communes ont été observées dans la presque totalité des cas, on est conduit à penser qu'une solution ne peut être bonne qu'à la condition de les posséder, et donc à n'explorer que le sous-espace qu'elles engendrent. Ceci permet d'accélérer considérablement les calculs, tout en assurant l'obtention d'une solution d'excellente qualité.

2.2. - APPLICATION AUX PROBLEMES D'ORDONNANCEMENT.

Les similitudes qui peuvent apparaître et être prises en compte dépendent bien évidemment du problème traité. Nous ne nous intéressons qu'à celui du voyageur de commerce. L'algorithme de résolution auquel nous avons couplé l'apprentissage des caractéristiques, est celui des distances.

Nous avons vu que pour un nombre de variables n donné, il est possible de trouver en fonction de R le nombre d'itérations minimal qui assure une probabilité de succès désirée. Soit k ce nombre.

Lorsque l'on observe les p premières de ces k solutions, on constate qu'elles ne diffèrent qu'en des endroits bien localisés sur la carte des villes à visiter. Autrement dit, elles ont en commun un nombre important d'arcs. Il est alors naturel d'admettre qu'ils ont de grandes chances d'appartenir à la solution optimale, et qu'en tout cas, ils définissent un sous-ensemble de bonnes solutions. C'est la raison pour laquelle leur sélection n'est plus remise en cause, et les itérations ultérieures sont faites de manière à ce qu'ils soient présents dans toutes les solutions produites. La taille du problème à résoudre est ainsi réduite et les temps de calcul accélérés. On pourra s'arrêter au bout des p itérations suivantes pour dégager les nouveaux arcs communs et restreindre encore le sous-ensemble des solutions à explorer. Celui-ci diminue donc au fur et à mesure que l'algorithme se déroule, jusqu'à ne plus comporter qu'un seul élément qui constitue la solution cherchée.

Dans la pratique, la mise en oeuvre de cette méthode pose quelques problèmes :

1 - Choix de la valeur de p : si on admet que les arcs sélectionnés doivent être présents dans les p solutions, ce choix résulte d'un compromis : en effet, plus la valeur de p est grande, moins le nombre d'arcs communs est élevé, mais ils ont plus de chances d'être optimaux. Ceci signifie en particulier que les temps de calcul croissent très rapidement avec p puisque la génération d'un grand nombre de solutions ne réduit que faiblement la taille du problème. En conséquence, la valeur de p devra être peu élevée. Mais il se peut qu'alors on sélectionne de mauvais arcs. Ce risque est en partie limité si la sélection ne porte que sur de bonnes solutions dont les arcs non optimaux sont peu nombreux. C'est la raison pour laquelle nous gènèrerons q solutions (q > p) parmi lesquelles nous ne garderons que les p meilleures pour y opérer la sélection

2 - Convergence de l'algorithme : Une fois que les arcs les plus évidents ont été choisis, il se peut qu'on arrive à une situation où aucun de ceux qui restent n'est vraiment caractéristique. L'algorithme boucle puisqu'alors aucune nouvelle sélection n'est opérée. Pour éviter cette impasse, dès que la taille du problème est suffisamment réduite, on explore r éléments appartenant au sous-ensemble auquel on a abouti et on conserve le meilleur comme solution définitive. Bien entendu, si au cours des diverses itérations, on a trouvé une solution de coût inférieur, c'est celle-ci qui est sauvegardée.

Le processus d'apprentissage ainsi défini a été utilisé pour résoudre en particulier le problème à 57 variables. On a dû se contenter d'appliquer l'algorithme des distances avec un rayon d'exploration égal à 2 puisque avec R=3, il faut à peu près cinq minutes pour effectuer une seule itération et le temps de génération des solutions aurait été beaucoup trop long. Par contre, dans la phase finale dans laquelle on entre lorsque le nombre d'arcs restant à sélectionner (NARC) devient inférieur ou égal à une certaine limite (LIMITE) les r solutions créées sont 3-optimales.

ETAPE 1 *Début*

NARC ← n

ETAPE 2 *Si* NARC ≤ LIMITE *alors aller à* Etape 3 ;

sinon itérer q fois l'algorithme des distances avec R=2,

retenir les p meilleures solutions ainsi obtenues.

sélectionner les arcs communs,

mettre à jour NARC,

aller à Etape 2.

ETAPE 3 effectuer r itérations de l'algorithme des distances avec R=3,

retenir la meilleure

la comparer à l'ensemble des solutions qui ont été générées,

conserver celle dont le coût est minimal

fin

Du choix des valeurs de p, q, r et LIMITE dépendent la qualité de la solution finale et la rapidité de la convergence. Un bon compromis entre ces deux objectifs est réalisé avec p=5, q=20, r=10, LIMITE=20.

2.3. - EVALUATION DES PERFORMANCES.

Dix essais ont été réalisés pour chacun des trois problèmes.

	33 VILLES									
ESSAI	1	2	3	4	5	6	7	8	9	10
SOLUTION	10861	10861	10861	10861	10861	10861	10861	10861	10861	10861
TEMPS	55,2	49,3	47,9	46,4	48,3	47,6	64,7	53,0	56,5	50,4

	42 VILLES									
ESSAI	1	2	3	4	5	6	7	8	9	10
SOLUTION	699	704	710	699	699	704	699	699	699	699
TEMPS	132,4	111,2	151,3	149,7	197,1	137,3	169,3	120,2	117,9	103,2

	57 VILLES									
ESSAI	1	2	3	4	5	6	7	8	9	10
SOLUTION	12978	13021	12966	13091	12985	12955	12978	12978	12985	13083
TEMPS	363,2	353,2	389,4	419,2	470,1	376,6	418,7	261,4	310,8	306,0

Qualité de la solution.

Elle est très satisfaisante puisque l'optimum a toujours été atteint au moins une fois, y compris pour le problème à 57 villes que la forme particulière de la matrice des distances en certaines régions rend peu commode à résoudre. Dans les autres cas, la fréquence observée de succès est très élevée (1,0 pour n = 33 ; 0,7 pour n = 42). Lorsque des solutions non optimales ont été trouvées, les coûts correspondants restent très proches de la valeur optimale.

Il est difficile d'évaluer a priori la probabilité de succès en fonction du nombre n des variables et des valeurs de p, q, r et LIMITE, car il faudrait connaître la probabilité de sélection d'un arc non-optimal à chaque étape qui est elle-même pratiquement impossible à appréhender. La seule chose que l'on puisse affirmer, c'est qu'elle est d'autant plus élevée que n est petit et que p,q et r sont grands.

Temps de calcul.

Là encore, l'évaluation de ce temps pose quelques problèmes puisqu'il est fonction essentiellement du nombre d'arcs sélectionnés à chaque étape qui dépend lui-même de n, de la structure de la matrice des distances et de la qualité des solutions générées. C'est la raison pour laquelle on peut constater qu'il est sujet à des fluctuations importantes pour une valeur de n donnée, dès lors qu'elle est largement supérieure à LIMITE. On peut en trouver simplement une borne inférieure qui est égale au temps nécessaire pour effectuer les q premières itérations lorsque NARC a sa valeur initiale. Cette borne ne reflète que très imparfaitement le temps de mise en oeuvre de l'algorithme qui peut être grossièrement approché par :

$$T \simeq 2\, n^3\, 10^{-3}$$

Si elle est valide au delà de n=57, cette formule montre que la méthode doit permettre de résoudre des problèmes à 100 variables en des temps de l'ordre de la demi-heure.

Quoiqu'il en soit, les temps observés sont comparables à ceux qui sont nécessaires pour n'appliquer qu'une seule fois l'algorithme des distances avec R=3, et la fréquence de succès est très largement supérieure aux probabilités à priori auxquelles aurait conduit cette unique itération.

3 - INTERET DE CES METHODES.

L'étude théorique des méthodes d'apprentissage demande un travail de longue haleine qui n'est pas réalisé ici, et nous avons dû parfois recourir à des solutions simplistes pour pouvoir les mettre en oeuvre. Notre but était d'en présenter les principes généraux et de montrer comment on peut les coupler à d'autres méthodes pour résoudre quelques cas concrets d'ordonnancement.

Nous avons pu observer que leur utilisation a toujours produit des solutions de très bonne qualité, et d'une manière générale sensiblement meilleures que celles fournies par les autres approches.

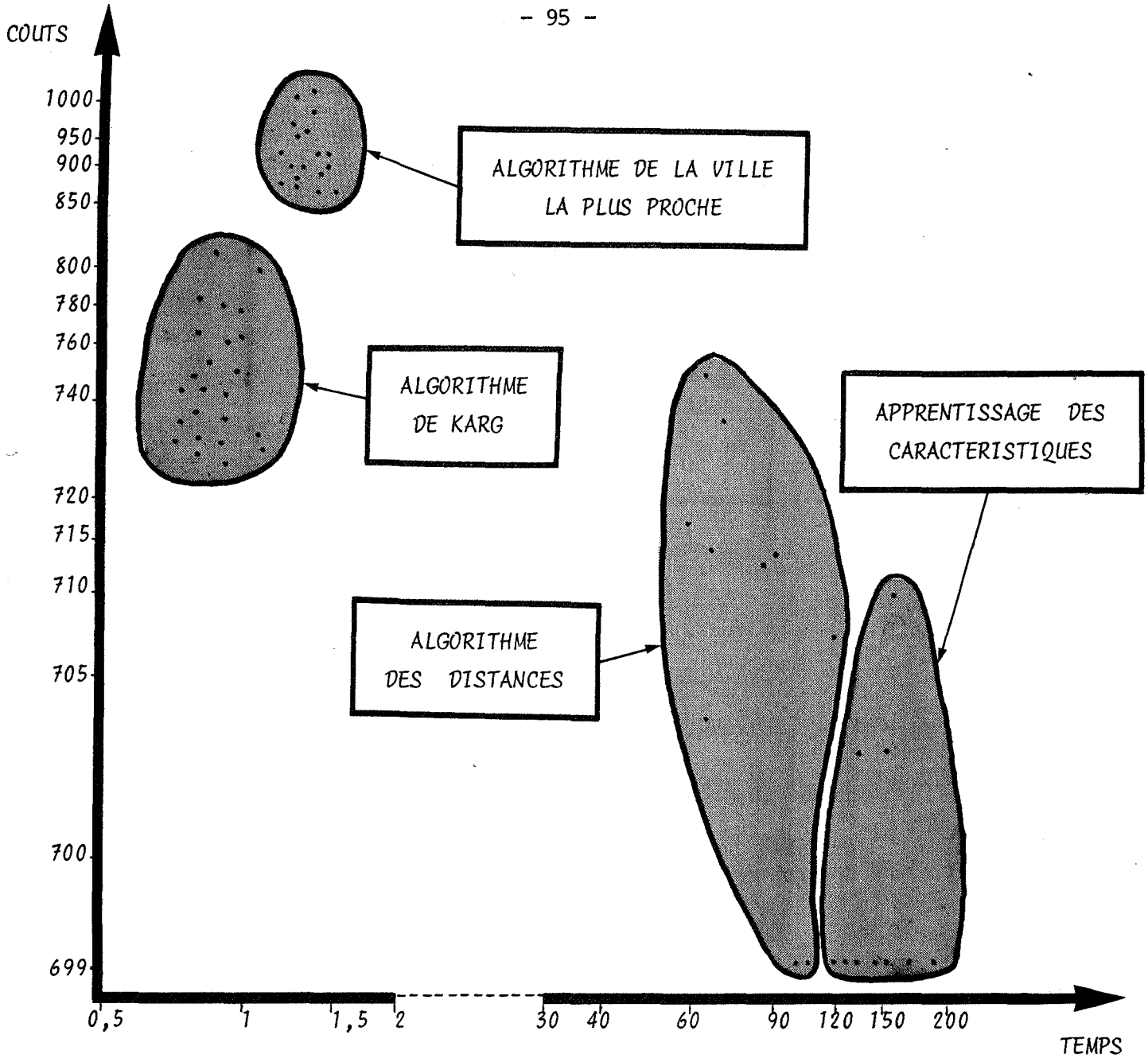
Pour ce qui est des durées de mise en oeuvre, on peut faire en sorte, par un choix judicieux des paramètres, qu'elles soient compatibles avec la contrainte temps réel. Ceci est d'autant plus vrai que les procédures peuvent être arrêtées quand on le désire, même si l'on n'a pas atteint le point de convergence.

CONCLUSION

CONCLUSION

La méthode la plus adaptée à un problème combinatoire temps réel est celle qui réalise le meilleur compromis entre la qualité de la solution désirée et le temps de calcul nécessaire à son obtention.

Il est donc a priori difficile de situer les algorithmes que nous avons présentés les uns par rapport aux autres, puisque d'une manière générale, l'amélioration du coût se fait au détriment du temps. C'est ce que nous avons pu constater tout au long de cette étude : exception faite des méthodes globales, nous sommes partis d'algorithmes rapides produisant des solutions de qualité moyenne (heuristiques non convergentes), pour finalement aboutir à des techniques plus longues, mais plus performantes en ce qui concerne le coût (heuristiques convergentes et apprentissage). Sur le graphique ci-contre sont représentés, en fonction du temps, les coûts des solutions produites par les divers algorithmes utilisés pour résoudre le problème du voyageur à 42 villes.



Cependant, les résultats auxquels nous sommes parvenus permettent de dégager deux des principales qualités que doivent posséder les méthodes de résolution envisageables pour avoir quelques chances d'être efficaces :

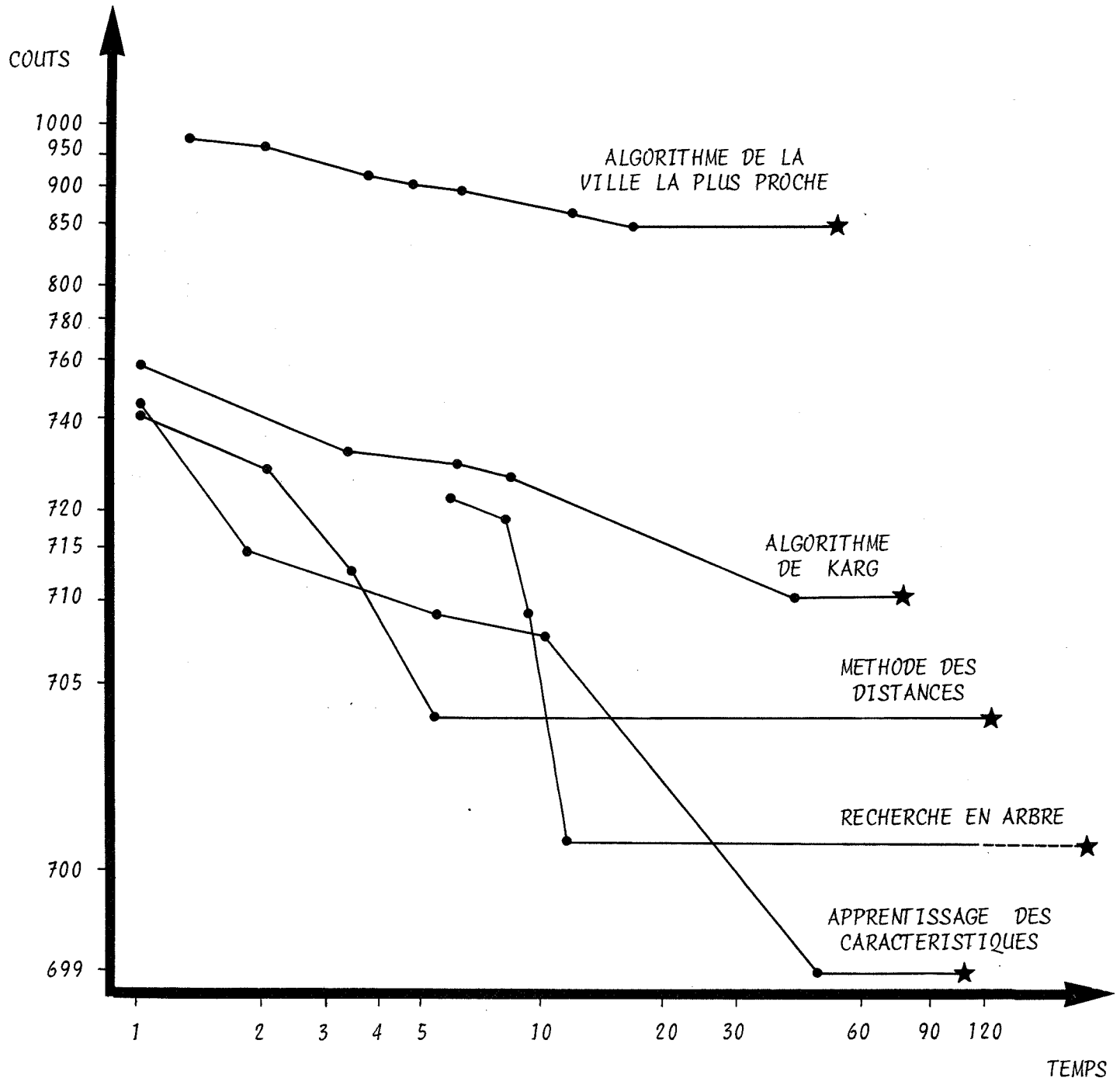
1 - *Une première solution réalisable doit être construite très rapidement*.

Cette condition est indispensable pour que la méthode puisse être utilisée même lorsque le délai de calcul alloué est très faible. A cet égard, toutes les méthodes proposées sont satisfaisantes, à l'exception de celles qui exigent, avant leur mise en oeuvre, de longues manipulations sur les données (c'est en particulier le cas des algorithmes de recherche en arbre : la première solution n'est obtenue que lorsque le tri des arcs a été effectué).

2 - *Cette première solution doit pouvoir être améliorée progressivement.*

Une fois que la première solution a été construite, et si le temps de calcul imparti le permet, il faut pouvoir procéder à des améliorations de cette solution. Ceci suppose que la méthode employée soit convergente, ou au moins itérable.

On a pu, en effet, remarquer que, chaque fois que de telles méthodes sont utilisées, les améliorations de la solution standard sont nombreuses dans les premières secondes de la mise en oeuvre de l'algorithme, puis de moins en moins fréquentes au fur et à mesure que l'on tend vers l'optimum. Cette constatation, assez évidente a priori, a été étayée par un calcul théorique au cours de la présentation de l'algorithme des distances. Elle est traduite par le graphique ci-dessous qui donne, en fonction du temps, le coût standard produit par chacune des méthodes proposées.



Il est donc essentiel de pouvoir utiliser la totalité du délai de calcul alloué, même s'il est bref, pour effectuer le plus grand nombre possible d'améliorations et obtenir ainsi une solution satisfaisante.

Dans ces conditions, les remarques que nous pouvons faire à propos des divers types de méthodes présentées sont les suivantes :

1 - HEURISTIQUES NON CONVERGENTES.

Il faut éviter de les utiliser seules, surtout lorsqu'elles ne sont pas itérables, à moins que l'une d'entre elles ne s'avère particulièrement adaptée au problème à résoudre.

Associées à d'autres méthodes, elles peuvent être d'un emploi intéressant pour construire au plus vite la première solution réalisable.

2 - HEURISTIQUES CONVERGENTES.

Elles sont toujours plus performantes que les précédentes. Comme nous l'avons montré, la longueur éventuelle de leur mise en oeuvre ne constitue pas un inconvénient majeur, mais bien plutôt une qualité, puisqu'elle permet de tirer parti de la totalité du temps de calcul alloué. Même si la procédure est arrêtée avant qu'on ait atteint le point de convergence, elle fournit toujours une bonne solution ; ceci est d'autant plus vrai qu'on a la possibilité de la coupler aux heuristiques non convergentes, ce qui assure que le résultat obtenu est au moins d'aussi bonne qualité que le meilleur de ceux qui sont produits par ces dernières.

La présente étude a montré que les méthodes classiques de recherche en arbre, qui sont globales lorsque l'on dispose d'un temps suffisant, conservent leur efficacité à condition de les adapter à la contrainte temps réel. Pour ce faire, il ne faut pas hésiter à effectuer certaines transformations, quitte à ne plus être certain de converger vers l'optimum : il est en effet beaucoup plus important de détecter de bonnes premières solutions, plutôt que de vouloir à tout prix tendre vers l'optimum, puisque de toutes les façons le point de convergence n'est en pratique jamais atteint.

Elle nous a également permis de mettre en évidence une classe d'algorithmes particulièrement intéressante ; ils sont caractérisés par une exploration limitée et dirigée de l'espace des solutions réalisables, et nous les avons regroupés sous le nom de méthode des distances. La remarque que nous venons de faire est encore vraie : leur utilisation n'est possible que si on limite le rayon maximal de la boule d'exploration, ce qui a pour conséquence de diminuer la probabilité théorique de détection de l'optimum.

3 - HEURISTIQUES D'APPRENTISSAGE.

Elles ont toujours été efficaces en ce qui concerne le coût de la solution produite.

Contrairement à ce qu'on aurait pu penser, leur mise en oeuvre est parfaitement envisageable en temps réel, puisqu'en agissant sur certains paramètres, on peut réduire à des proportions raisonnables le temps d'exécution : de plus, il est possible d'en arrêter le déroulement quand on le désire. Par ailleurs, on a pu constater que leur couplage à une méthode convergente en a accéléré de manière significative le temps de réponse.

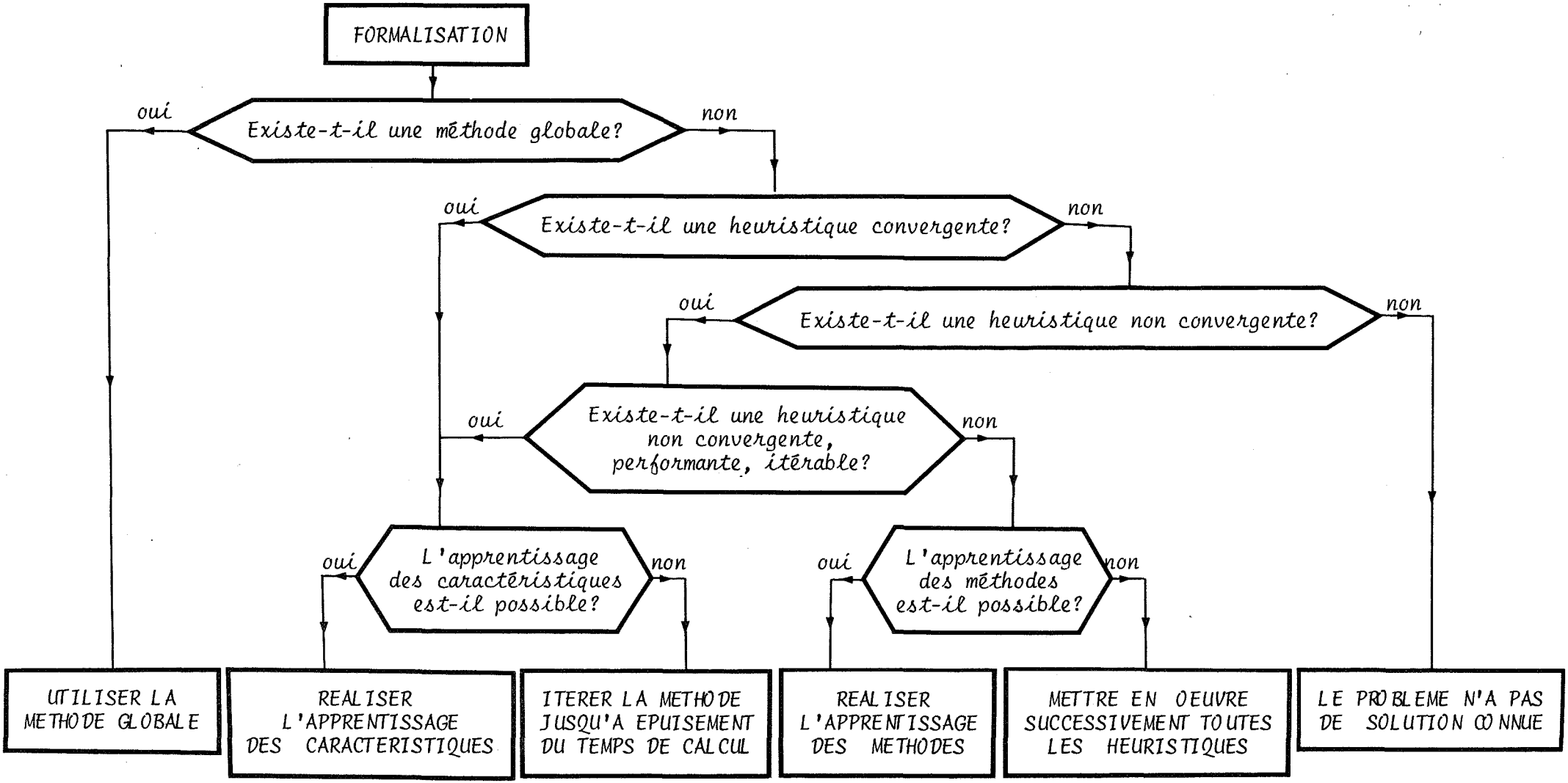
4 - METHODES GLOBALES.

Elles doivent bien évidemment être préférées à toutes les autres dans les quelques cas où elles peuvent être utilisées.

Si donc plusieurs approches sont envisageables pour résoudre un problème donné, les remarques que nous venons de faire permettent de déterminer a priori un ordre préférentiel entre elles qui est traduit dans l'arbre de décision qui figure à la page suivante.

Bien que nous les ayons présentées indépendamment les unes des autres, les différentes méthodes sont susceptibles d'être couplées entre elles en vue d'une utilisation simultanée, en particulier dès qu'on fait appel à l'apprentissage.

C'est la raison pour laquelle les progrès futurs que l'on pourra faire en matière de résolution de problèmes combinatoires en temps réel ne nous semblent pas devoir résider dans la mise au point d'heuristiques sophistiquées suffisamment générales pour pouvoir être utilisées dans de nombreux cas ; les problèmes sont trop divers pour pouvoir être tous abordés et résolus de manière satisfaisante de la même façon.


FORMALISATION
Existe-t-il une méthode globale?
oui
non
Existe-t-il une heuristique convergente?
oui
non
Existe-t-il une heuristique non convergente?
oui
non
Existe-t-il une heuristique non convergente, performante, itérable?
oui
non
L'apprentissage des caractéristiques est-il possible?
oui
non
L'apprentissage des méthodes est-il possible?
oui
non
UTILISER LA METHODE GLOBALE
REALISER L'APPRENTISSAGE DES CARACTERISTIQUES
ITERER LA METHODE JUSQU'A EPUISEMENT DU TEMPS DE CALCUL
REALISER L'APPRENTISSAGE DES METHODES
METTRE EN OEUVRE SUCCESSIVEMENT TOUTES LES HEURISTIQUES
LE PROBLEME N'A PAS DE SOLUTION CONNUE

Par contre, il est certain que pour chacun d'eux, on pourra trouver quelques recettes simples, convergentes ou non, qui tiennent compte de leur spécificité, et dont l'association avec des techniques d'apprentissage ne peut qu'améliorer les performances. C'est donc sur ce dernier type de méthodes que doivent porter les efforts, d'une part pour définir les modalités de leur mise en oeuvre, mais surtout pour préciser les notions auxquelles elles font appel.

L'apprentissage des caractéristiques ne semble pas devoir présenter de problèmes majeurs, dès lors qu'on sait quelles caractéristiques extraire.

En ce qui concerne l'apprentissage des méthodes, la principale difficulté est de savoir comment définir les situations de prise de décision. Si elle était résolue, il serait probablement assez facile de déterminer, une fois pour toutes, par un apprentissage approprié, l'heuristique la plus adaptée à chaque situation. La résolution du problème en temps réel serait alors considérablement simplifiée puisqu'il suffirait d'utiliser ces heuristiques, sans remettre en cause l'apprentissage qui a dicté leur choix.

ANNEXES

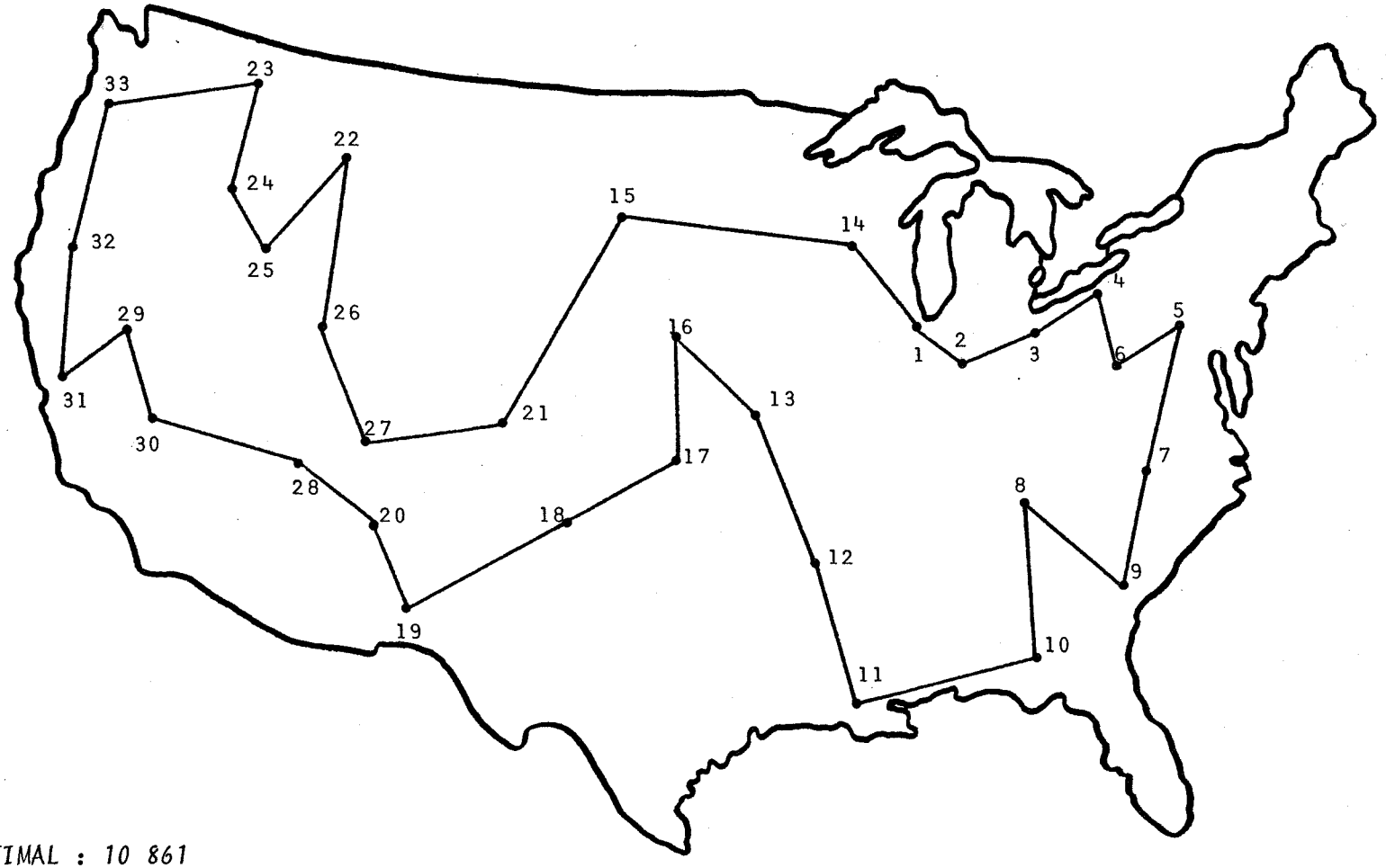
ANNEXE 1

PROBLEME A 33 VILLES
****

1	0																							
2	184	0																						
3	292	195	0																					
4	449	310	215	0																				
5	670	540	380	288	0																			
6	516	357	232	200	211	0																		
7	598	514	434	506	436	381	0																	
8	618	434	493	787	514	642	295	0																
9	881	697	719	790	632	697	224	320	0															
10	909	954	958	1020	974	952	541	341	318	0														
11	978	892	1031	1246	1352	1180	843	538	747	441	0													
12	654	597	803	1018	1154	1104	766	461	749	634	380	0												
13	504	503	722	937	1043	806	986	722	1042	954	784	404	0											
14	276	460	568	725	946	817	874	894	1214	1185	1218	660	452	0										
15	780	964	1072	1229	1450	1321	1378	1326	1646	1672	1410	1030	626	476	0									
16	529	644	789	1004	1184	1001	1214	950	1270	1213	1043	632	219	436	419	0								
17	805	698	917	1132	1238	1055	1113	812	1162	1027	779	473	195	637	634	256	0							
18	1181	1007	1226	1441	1547	1364	1375	1080	1134	1138	783	611	563	1046	759	624	368	0						
19	1548	1444	1630	1845	1984	1801	1726	1431	1685	1477	1134	1033	906	1389	1094	967	711	404	0					
20	1547	1454	1668	1883	1994	1811	1879	1584	1776	1632	1267	1053	944	1427	1196	1005	749	442	251	0				
21	1239	1167	1353	1568	1707	1524	1584	1313	1633	1498	1151	979	614	988	600	525	471	368	512	507	0			
22	1538	1733	1930	2145	2209	2090	2136	2078	2398	2332	2110	1782	1378	1300	760	1229	1382	1319	1163	930	910	0		
23	1999	2158	2291	2448	2669	2515	2597	2500	2820	2675	2336	2164	1707	1860	1375	1582	1658	1545	1389	1156	1237	436	0	
24	1716	1875	2009	2165	2385	2232	2488	2217	2537	2392	2053	1881	1422	1577	1106	1244	1375	1262	1106	873	954	483	283	0
25	1580	1738	1872	2029	2250	2095	2352	2081	2401	2256	1917	1745	1286	1473	988	1147	1239	1126	970	737	818	379	419	136
26	1425	1569	1717	1874	2109	1926	2115	1844	2164	2019	1680	1508	1118	1335	802	913	1002	889	733	500	581	430	656	373
27	1560	1549	1852	2009	2089	1906	2063	1792	2112	1967	1456	1274	1032	1540	1068	1007	944	665	521	282	491	768	994	711
28	1918	1744	1963	2178	2284	2101	2174	1879	2071	1892	1562	1348	1239	1722	1258	1300	1044	737	526	295	802	816	1022	739
29	2065	2102	2357	2514	2642	2459	2626	2355	2675	2530	2191	2019	1673	1905	1432	1570	1507	1320	1109	878	1124	842	715	432
30	2284	2131	2326	2441	2671	2488	2418	2123	2437	2259	1929	1715	1606	1924	1451	1589	1411	1104	893	662	1143	1004	981	698
31	2340	2348	2543	2658	2888	2705	2869	2598	2918	2773	2434	2262	1916	2148	1675	1813	1750	1468	1102	871	1367	1085	958	675
32	2247	2327	2539	2696	2867	2684	2851	2580	2900	2755	2416	2244	1898	2130	1657	1795	1732	1545	1334	1103	1349	1014	814	531
33	2163	2322	2455	2612	2833	2679	2761	2664	2984	2839	2500	2328	1871	2024	1539	1746	1822	1709	1553	1320	1391	693	346	447
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

25	0								
26	237	0							
27	575	358	0						
28	603	386	545	0					
29	465	533	840	739	0				
30	599	589	768	523	256	0			
31	1033	778	1092	982	243	349	0		
32	1015	760	1047	964	225	497	266	0	
33	583	820	1158	1355	581	847	710	444	0
	25	26	27	28	29	30	31	32	33

33 VILLES



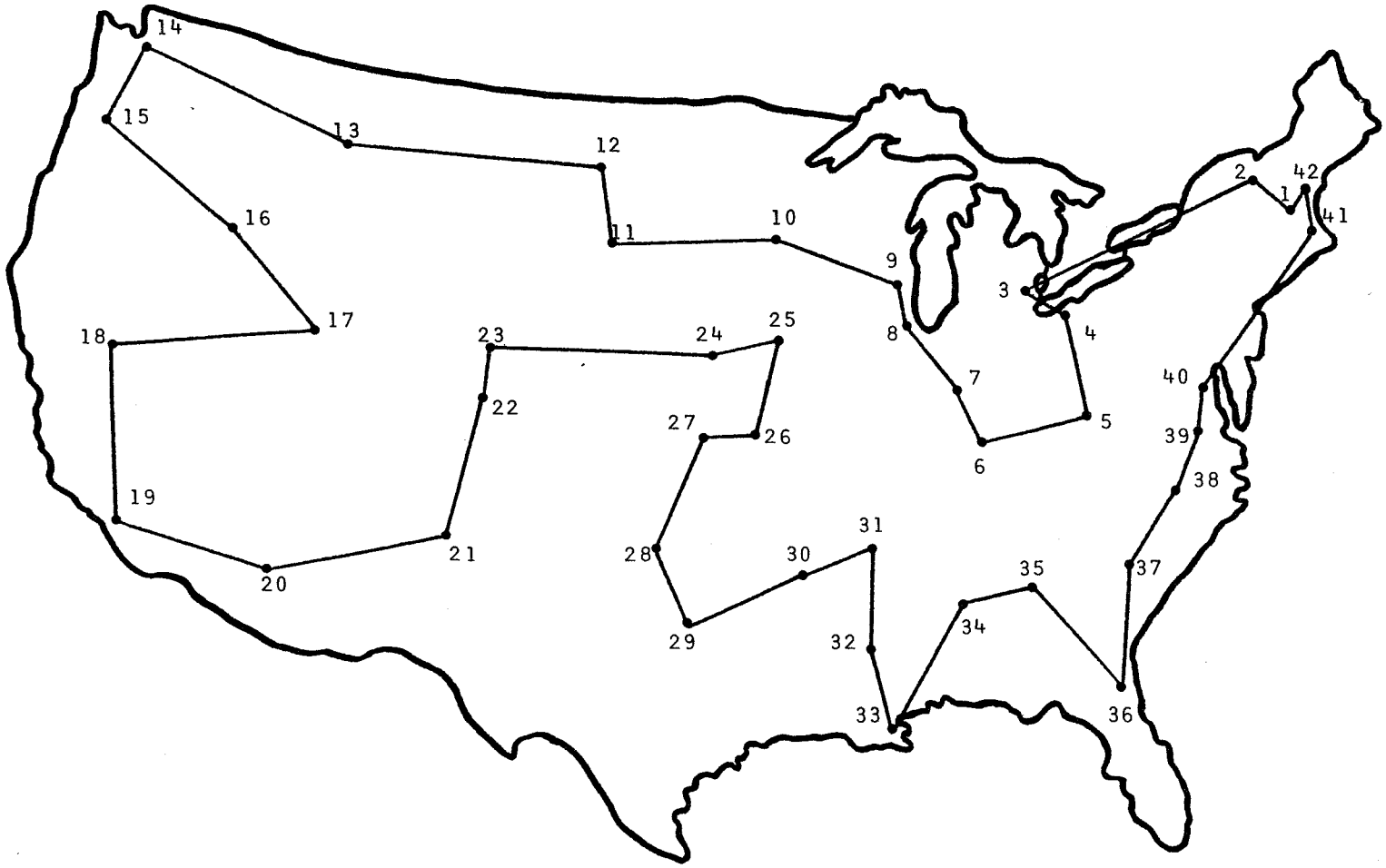
CIRCUIT OPTIMAL : 10 861

PROBLEME A 42 VILLES
****

1	0																														
2	8	0																													
3	39	45	0																												
4	37	47	9	0																											
5	50	49	21	15	0																										
6	61	62	21	20	17	0																									
7	58	60	16	17	18	6	0																								
8	59	60	15	20	26	17	10	0																							
9	62	66	20	25	31	22	15	5	0																						
10	81	81	40	44	50	41	35	24	20	0																					
11	103	107	62	67	72	63	57	46	41	23	0																				
12	108	117	66	71	77	68	61	51	46	26	11	0																			
13	145	149	104	108	114	106	99	88	84	63	49	40	0																		
14	181	185	140	144	150	142	135	124	120	99	85	76	35	0																	
15	187	191	146	150	156	142	137	130	125	105	90	81	41	10	0																
16	161	170	120	124	130	115	110	104	105	90	72	64	34	31	27	0															
17	142	146	101	104	111	97	91	85	86	75	51	59	29	53	48	21	0														
18	174	178	133	138	143	129	123	117	118	107	83	84	54	46	35	26	31	0													
19	185	186	142	145	140	130	126	124	128	118	93	101	72	69	58	58	43	26	0												
20	164	165	120	123	124	106	106	105	110	104	86	97	71	93	82	62	42	45	22	0											
21	137	139	94	96	94	80	78	77	84	77	56	64	65	90	87	58	36	68	50	30	0										
22	117	122	77	80	83	68	62	60	61	50	34	42	49	82	77	60	30	62	70	49	21	0									
23	114	118	73	78	84	69	63	57	59	48	28	36	43	77	72	45	27	59	69	55	27	5	0								
24	85	89	44	48	53	41	34	28	29	22	23	35	69	105	102	74	56	88	99	81	54	32	29	0							
25	77	80	36	40	46	34	27	19	21	14	29	40	77	114	111	84	64	96	107	87	60	40	37	8	0						
26	87	89	44	46	46	36	28	29	32	27	36	47	78	116	112	84	66	98	95	75	47	36	39	12	11	0					
27	91	93	48	50	48	34	32	33	36	30	34	45	77	115	110	83	63	97	91	72	44	32	36	9	15	3	0				
28	105	106	62	63	64	47	46	49	54	48	46	59	85	119	115	88	66	98	79	59	31	36	42	28	33	21	20	0			
29	111	113	69	71	66	51	53	56	61	57	59	71	96	130	126	98	75	98	85	62	38	47	53	39	42	29	30	12	0		
30	91	92	50	51	46	30	34	38	43	49	60	71	103	141	136	109	90	115	99	81	53	61	62	36	34	24	28	20	20	0	
31	83	85	42	43	38	22	26	32	36	51	63	75	106	142	140	112	93	126	108	88	60	64	66	39	36	27	31	28	28	8	
32	89	91	55	55	50	34	39	44	49	63	76	87	120	155	150	123	100	123	100	86	62	71	78	52	49	39	44	35	24	15	
33	95	97	64	63	56	42	49	56	60	75	86	97	126	160	155	128	104	128	113	90	67	76	82	62	59	49	53	40	29	25	
34	74	81	44	43	35	23	30	39	44	62	78	89	121	159	155	127	108	136	124	101	75	79	81	54	50	42	46	43	39	23	
35	67	69	42	41	31	25	32	41	46	64	83	90	130	164	160	133	114	146	134	111	85	84	86	59	52	47	51	53	49	32	
36	74	76	61	60	42	44	51	60	66	83	102	110	147	185	179	155	133	159	146	122	98	105	107	79	71	66	70	70	60	48	
37	57	59	46	41	25	30	36	47	52	71	93	98	136	172	172	148	126	158	147	124	121	97	99	71	65	59	63	67	62	46	
38	45	46	41	34	20	34	38	48	53	73	96	99	137	176	178	151	131	163	159	135	108	102	103	73	67	64	69	75	72	54	
39	35	37	35	26	18	34	36	46	51	70	93	97	134	171	176	151	129	161	163	139	118	102	101	71	65	65	70	84	78	58	
40	29	33	30	21	18	35	33	40	45	65	87	91	117	166	171	144	125	157	156	139	113	95	97	67	60	62	67	79	82	62	
41	3	11	41	37	47	57	55	58	63	83	105	109	147	186	188	164	144	176	182	161	134	119	116	86	78	84	88	101	108	88	
42	5	12	55	41	53	64	61	61	66	84	111	113	150	186	192	166	147	180	188	167	140	124	119	90	87	90	94	107	114	77	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	

31	0											
32	12	0										
33	23	11	0									
34	14	14	21	0								
35	24	24	30	9	0							
36	40	36	33	25	18	0						
37	38	37	43	23	13	17	0					
38	46	40	54	34	24	29	12	0				
39	50	56	62	41	32	39	21	9	0			
40	53	59	66	45	38	45	27	15	6	0		
41	80	86	92	71	64	71	54	41	32	25	0	
42	86	92	98	80	74	77	60	48	38	32	6	0
	31	32	33	34	35	36	37	38	39	40	41	42

42 VILLES



CIRCUIT OPTIMAL : 699

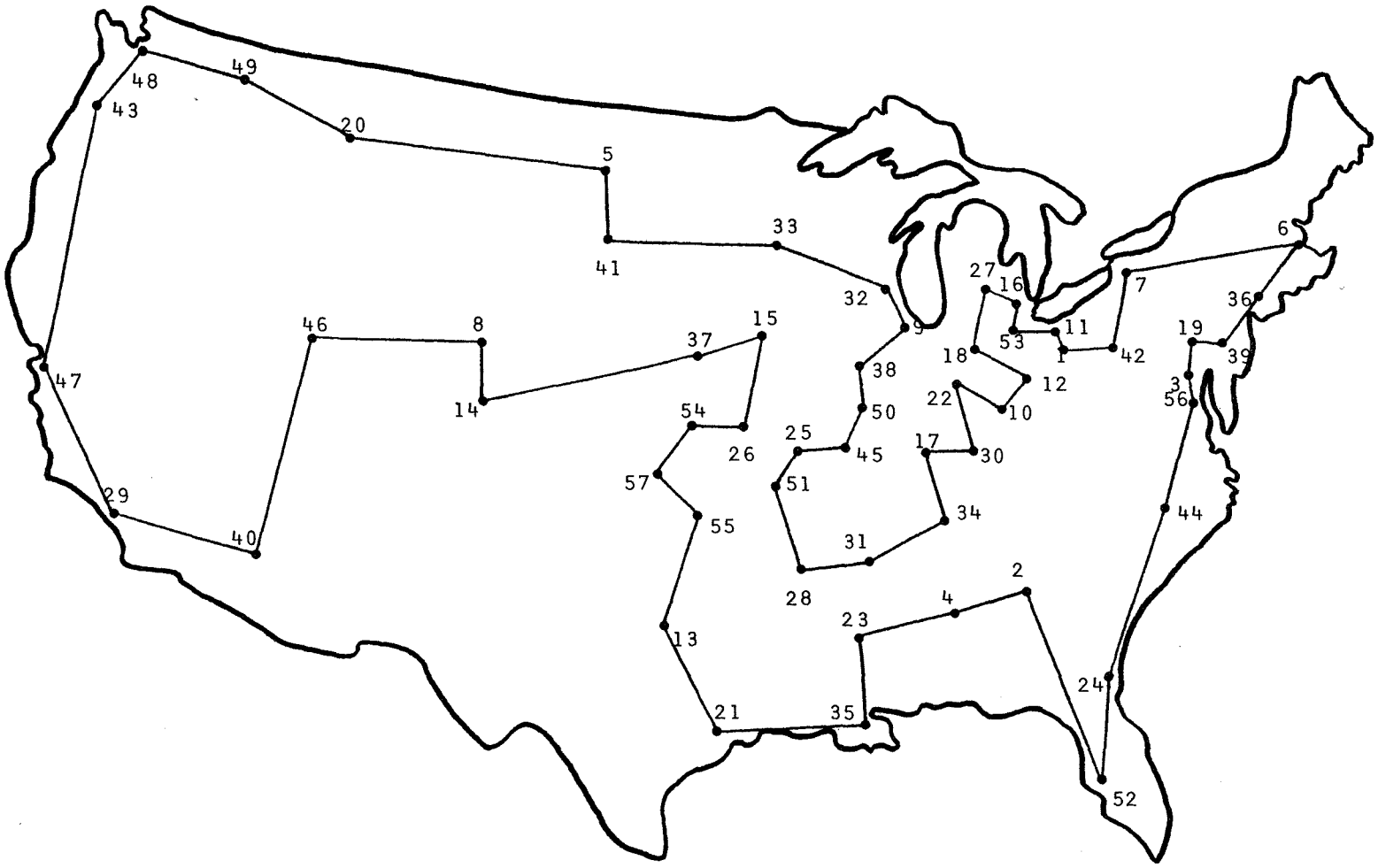
PROBLEME A 57 VILLES
****

1	0																						
2	678	0																					
3	331	635	0																				
4	702	152	791	0																			
5	1210	1523	1531	1475	0																		
6	655	1074	399	1190	1818	0																	
7	210	881	364	900	1377	447	0																
8	1334	1457	1655	1365	580	1942	1497	0															
9	364	702	686	653	853	994	527	968	0														
10	227	454	498	475	1187	869	425	1188	324	0													
11	33	695	351	714	1187	631	186	1311	341	239	0												
12	121	554	395	584	1205	763	327	1248	359	106	140	0											
13	1180	820	1422	665	1172	1821	1378	857	937	953	1254	1054	0										
14	1380	1411	1631	1302	681	1997	1507	101	1013	1172	1366	1236	784	0									
15	706	899	1027	823	672	1314	869	628	339	576	688	701	701	674	0								
16	190	709	511	736	1149	699	252	1263	293	255	167	185	1203	1318	635	0							
17	452	414	723	365	1114	1107	649	1076	292	225	463	331	749	1030	505	442	0						
18	202	603	556	591	1021	843	398	1099	175	151	212	154	997	1149	471	158	284	0					
19	301	729	480	816	1501	398	284	1625	655	461	321	382	1436	1619	997	481	713	526	0				
20	1806	2081	2127	2021	631	2414	1969	712	1449	1783	1783	1801	1577	811	1240	1735	1720	1617	2097	0			
21	1283	842	1457	665	1414	1856	1491	1109	1091	1056	1295	1162	242	1028	943	1311	843	1128	1482	1819	0		
22	291	510	568	487	1032	936	481	1079	184	109	295	173	881	1063	467	274	168	116	556	1629	1011	0	
23	913	402	1038	247	1453	1437	1102	1278	753	677	916	783	411	1195	818	924	495	763	1063	1972	429	647	0
24	929	316	793	430	1839	1191	1083	1773	1027	776	960	859	1007	1727	1224	1031	739	927	865	2429	907	835	596	0
25	656	680	934	586	928	1338	896	801	393	468	710	538	562	755	271	634	298	476	921	1457	785	365	563	996
26	776	805	1053	713	789	1454	1009	652	504	596	823	658	495	606	206	775	424	587	1040	1308	737	485	664	1121
27	237	731	558	721	1089	845	405	1209	239	270	219	232	1146	1261	578	84	414	130	528	1681	1252	246	893	1046
28	847	521	1089	393	1193	1488	1045	1049	650	620	859	735	333	965	576	871	416	681	1075	1712	441	565	260	822
29	2371	2215	2649	2063	1659	3042	2597	1169	2092	2182	2411	2246	1405	1157	1794	2363	2012	2175	2628	1234	1547	2073	1816	2415
30	338	396	609	373	1146	980	536	1170	298	111	351	218	851	1124	572	367	125	218	572	1743	945	114	575	721
31	717	382	950	254	1248	1350	906	1111	538	481	720	596	472	1048	605	719	277	561	936	1767	566	445	213	679
32	452	793	773	739	771	1060	615	1003	91	429	429	447	1010	1002	356	381	378	263	743	1367	1164	271	830	1109
33	769	1080	1096	1037	430	1377	932	800	412	746	746	764	953	846	253	698	687	580	1060	1027	1195	592	1051	1396
34	513	257	728	205	1205	1162	718	1206	448	293	532	398	694	1160	629	541	160	400	738	1862	788	297	403	573
35	1043	518	1145	354	1623	1544	1241	1369	930	816	1055	928	503	1287	1006	1071	669	943	1170	2076	358	827	187	565
36	487	862	187	978	1691	216	445	1811	841	653	507	551	1605	1866	1183	667	882	712	186	2273	1644	724	1225	980
37	841	1014	1142	917	584	1449	1004	493	474	698	818	755	663	539	135	770	621	606	1132	1072	905	589	806	1325
38	451	656	766	617	858	1102	657	869	155	323	471	380	805	910	253	423	260	242	785	1456	973	214	661	972
39	406	772	97	888	1610	304	366	1730	760	572	426	470	1524	1785	1102	586	801	631	105	2202	1554	643	1135	890
40	2001	1833	2280	1678	1500	2682	2237	920	1732	1824	2051	1886	1029	819	1441	2003	1652	1815	2268	1183	1156	1713	1440	2036
41	1127	1381	1448	1307	207	1745	1290	444	771	1104	1104	1122	965	522	492	1056	997	938	1418	739	1207	951	1238	1697
42	105	714	230	751	1309	598	216	1429	459	284	125	182	1223	1484	801	285	509	330	200	1901	1326	356	956	897
43	2492	2632	2813	2572	1317	3100	2655	1207	2131	2395	2469	2455	2057	1285	1835	2421	2283	2303	2783	685	2235	2286	2468	2980
44	594	402	301	559	1794	701	616	1696	948	521	614	493	1224	1592	1090	774	671	538	376	2390	1225	629	806	492
45	529	554	806	490	987	1239	797	906	294	340	611	411	643	860	335	512	170	354	794	1562	799	238	502	870
46	1792	1925	2113	1796	953	2400	1955	458	1425	1646	1769	1706	1253	507	1086	1721	1533	1257	2083	498	1428	1537	1664	2214
47	2554	2543	2875	2391	1646	3162	2717	1220	2187	2408	2531	2468	1763	1264	1848	2483	2295	2319	2845	1137	1946	2300	2174	2779
48	2416	2691	2737	2631	1223	3024	2579	1279	2055	2393	2393	2411	2129	1357	1850	2345	2330	2227	2707	610	2307	2239	2540	3007
49	2128	2403	2440	2343	935	2736	2291	1023	1767	2105	2105	2123	1885	1101	1502	2057	2042	1939	2419	322	2141	1951	2252	2719
50	475	603	760	563	921	1138	695	902	193	301	510	365	744	873	293	459	212	276	821	1507	900	192	588	919
51	750	665	1027	538	964	1460	1018	827	515	563	832	632	422	764	347	733	391	579	1015	1483	645	459	489	968
52	1106	467	984	547	1991	1379	1260	1903	1169	921	1137	1022	1090	1833	1366	1177	881	1072	1041	2555	980	978	679	200
53	133	652	454	673	1089	741	296	1209	239	198	110	128	1149	1264	581	57	388	110	424	1681	1231	220	867	924
54	839	868	1120	773	751	1517	1072	597	567	659	886	721	517	538	268	838	487	654	1103	1262	738	552	727	1184
55	931	809	1208	683	972	1641	1199	770	696	744	1013	813	268	686	463	914	572	760	1196	1440	510	640	550	1112
56	331	637	38	753	1531	437	361	1655	685	492	351	395	1384	1710	1027	511	717	556	112	2127	1419	568	1000	755
57	971	922	1248	793	739	1649	1204	595	699	791	1018	853	383	508	401	970	619	782	1235	1256	625	698	720	1211
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

25	0																							
26	149	0																						
27	573	684	0																					
28	366	404	811	0																				
29	1735	1588	2272	1695	0																			
30	392	518	348	518	2106	0																		
31	350	459	691	139	1825	379	0																	
32	465	550	327	723	2138	387	618	0																
33	509	458	644	829	1942	706	838	330	0															
34	423	554	530	361	2047	182	222	534	847	0														
35	760	834	1060	430	1905	712	400	1017	1238	530	0													
36	1089	1209	714	1273	2797	764	1134	929	1246	915	1332	0												
37	353	204	713	608	1669	694	663	491	359	751	1038	1318	0											
38	243	352	369	530	1940	326	448	227	423	407	842	971	375	0										
39	1009	1128	633	1193	2716	682	1054	848	1165	825	1242	92	1237	890	0									
40	1375	1228	1912	1336	390	1745	1467	1778	1657	1689	1514	2437	1313	1580	2358	0								
41	743	594	1002	998	1556	1065	1053	700	393	1121	1415	1504	390	745	1528	1341	0							
42	720	840	332	809	2429	395	760	547	864	562	1092	386	936	589	305	2069	1222	0						
43	2008	1859	2367	2250	994	2377	2318	2053	1713	2413	2560	2959	1700	2076	2888	1273	1425	2587	0					
44	960	1086	821	897	2583	568	758	1036	1353	559	919	488	1213	838	398	2225	1711	451	3076	0				
45	128	254	503	359	1842	264	289	367	549	300	689	962	453	174	885	1482	827	596	2113	832	0			
46	1260	1109	1667	1456	731	1627	1535	1460	1231	1663	1756	2269	951	1327	2188	685	845	1887	807	2154	1363	0		
47	2023	1871	2429	2023	405	2389	2153	2222	1993	2375	2264	3031	1713	2090	2950	791	1607	2649	669	2916	2125	762	0	
48	2067	1918	2291	2322	1170	2353	2377	1977	1637	2472	2632	2883	1682	2066	2812	1490	1349	2511	176	3000	2172	879	845	0
49	1779	1630	2003	2034	1246	2065	2059	1689	1349	2184	2398	2595	1394	1778	2524	1368	1061	2223	364	2712	1884	721	922	288
50	200	311	373	457	1899	279	375	265	486	359	769	1007	415	73	926	1539	785	548	2109	821	101	1360	2122	2117
51	140	175	724	229	1634	485	284	588	600	447	659	1183	379	383	1102	1276	769	903	2034	1003	221	1251	1961	2093
52	1142	1244	1198	924	2489	864	785	1259	1547	724	646	1170	1448	1123	1080	2111	1838	1080	3103	662	1021	2340	2831	3165
53	586	693	104	755	2282	310	665	327	644	484	1014	610	716	369	529	1979	1002	228	2416	621	506	1667	2429	2291
54	208	63	747	467	1532	585	522	612	521	626	902	1272	167	415	1191	1172	557	903	1801	1149	317	1049	1811	1872
55	321	257	905	290	1448	666	429	769	715	628	689	1364	395	564	1283	1093	765	996	1970	1187	402	1171	1776	2042
56	934	1053	558	1051	2649	598	912	773	1090	690	1107	225	1163	815	135	2289	1446	230	2813	263	806	2113	2875	2737
57	344	195	879	460	1393	713	532	745	653	703	834	1404	306	547	1323	1033	582	1036	1797	1251	449	1003	1721	1866
	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48

49	0								
50	1829	0							
51	1805	322	0						
52	2877	1070	1069	0					
53	2003	405	727	1120	0				
54	1584	374	235	1307	256	0			
55	1762	503	181	1214	908	228	0		
56	2449	851	1027	945	454	1116	1222	0	
57	1578	506	248	1317	888	139	183	1448	0
	49	50	51	52	53	54	55	56	57

57 VILLES



MEILLEUR CIRCUIT CONNU : 12 955

ANNEXE 2

PORBLEME 6 x 6

	PIECE 1		PIECE 2		PIECE 3		PIECE 4		PIECE 5		PIECE 6	
N°TACHE	N°MACH	TEMPS	N°MACH	TEMPS	N°MACH	TEMPS	N°MACH	TEMPS	N°MACH	TEMPS	N°MACH	TEMPS
1	3	1	2	8	3	5	2	5	3	9	2	3
2	1	3	3	5	4	4	1	5	2	3	4	3
3	2	6	5	10	6	8	3	5	5	5	6	9
4	4	7	6	10	1	9	4	3	6	4	1	10
5	6	3	1	10	2	1	5	8	1	3	5	4
6	5	6	4	4	5	1	6	9	4	1	3	1

PROBLEME 10 x 10

	PIECE 1		PIECE 2		PIECE 3		PIECE 4		PIECE 5	
N° TACHE	N° MACH	TEMPS	N° MACH	TEMPS	N° MACH	TEMPS	N° MACH	TEMPS	N° MACH	TEMPS
1	1	29	1	43	2	91	2	81	3	14
2	2	78	3	90	1	85	3	95	1	6
3	3	9	5	75	4	39	1	71	2	22
4	4	36	10	11	3	74	5	99	6	61
5	5	49	4	69	9	90	7	9	4	26
6	6	11	2	28	6	10	9	52	5	69
7	7	62	7	46	8	12	8	85	9	21
8	8	56	6	46	7	89	4	98	8	49
9	9	44	8	72	10	45	10	22	10	72
10	10	21	9	30	5	33	6	43	7	53

	PIECE 6		PIECE 7		PIECE 8		PIECE 9		PIECE 10	
N°TACHE	N° MACH	TEMPS	N°MACH	TEMPS	N° MACH	TEMPS	N°MACH	TEMPS	N°MACH	TEMPS
1	3	84	2	46	3	31	1	76	2	85
2	2	2	1	37	1	86	2	79	1	13
3	6	52	4	61	2	46	4	76	3	61
4	4	95	3	13	6	74	6	51	7	7
5	9	48	7	32	5	32	3	85	9	64
6	10	72	6	21	7	88	10	11	10	76
7	1	47	10	32	9	19	7	40	6	47
8	7	65	9	89	10	48	8	89	4	52
9	5	6	8	30	8	36	5	26	5	90
10	8	25	5	55	4	79	9	74	8	45

PROBLEME 20 x 5

	PIECE 1		PIECE 2		PIECE 3		PIECE 4		PIECE 5	
N°TACHE	N° MACH	TEMPS	N°MACH	TEMPS	N° MACH	TEMPS	N° MACH	TEMPS	N° MACH	TEMPS
1	1	29	1	43	2	91	2	81	3	14
2	2	9	2	75	1	39	1	71	2	22
3	3	49	4	69	3	90	5	9	1	26
4	4	62	3	46	5	12	3	85	4	21
5	5	44	5	72	4	45	4	22	5	72

	PIECE 6		PIECE 7		PIECE 8		PIECE 9		PIECE 10	
N°TACHE	N°MACH	TEMPS	N°MACH	TEMPS	N°MACH	TEMPS	N°MACH	TEMPS	N°MACH	TEMPS
1	3	84	2	46	3	31	1	76	2	85
2	2	52	1	61	2	46	4	76	3	61
3	5	48	3	32	1	32	3	85	1	64
4	1	47	4	32	4	19	2	40	4	47
5	4	6	5	30	5	36	5	26	5	90

	PIECE 11		PIECE 12		PIECE 13		PIECE 14		PIECE 15	
N° TACHE	N°MACH	TEMPS	N° MACH	TEMPS	N° MACH	TEMPS	N° MACH	TEMPS	N° MACH	TEMPS
1	2	78	3	90	1	85	3	95	1	6
2	4	36	1	11	3	74	1	99	2	61
3	1	11	2	28	2	10	2	52	5	69
4	5	56	4	46	4	89	4	98	3	49
5	3	21	5	30	5	33	5	43	4	53

	PIECE 16		PIECE 17		PIECE 18		PIECE 19		PIECE 20	
N° TACHE	N°MACH	TEMPS	N°MACH	TEMPS	N°MACH	TEMPS	N° MACH	TEMPS	N° MACH	TEMPS
1	2	2	1	37	1	86	2	69	1	13
2	1	95	3	13	2	74	3	51	2	7
3	4	72	2	21	5	88	1	11	3	76
4	5	65	4	89	3	48	4	89	4	52
5	3	25	5	55	4	79	5	74	5	45

VU : Le Président de la thèse,

VU et permis d'imprimer :
TOULOUSE, le

LE PRESIDENT
de l'UNIVERSITE des SCIENCES SOCIALES
DE TOULOUSE,

R. PALLARD

B I B L I O G R A P H I E

[1] S. ASHOUR - A Branch and Bound approach to the job-shop scheduling problem.
International journal for production research, vol 11 N° 1 - 1973

[2] R.W CONWAY, W.L. MAXWELL, L.W. MILLER - Theory of scheduling.
Addison Wesley - 1967

[3] H.H. GREENBERG - Solution of the general scheduling problem.
Operations research, 16 - 1968.

[4] J.R. JACKSON - Scheduling a production line to minimize maximum tardiness.
Research report 43, Management sciences research project, UCLA - 1955.

[5] J.R. JACKSON - An extension of Johnson's results on job-lot scheduling.
Nav. res. log. quart 3, N° 3 - 1956.

[6] S.M. JOHNSON - Optimal two and three stage production schedules.
Nav. res. log. quart 1, N° 1 - 1954.

[7] R.L. KARG - A heuristic approach to solving travelling salesman problems.
Management science, vol 10 N° 2 - 1964.

[8] J. LABETOULLE - Un algorithme optimal pour la gestion des processus en temps réel.
RAIRO, B-1 - 1974.

[9] S. LIN - Computer solutions of the travelling salesman problem.
Bell system technical journal 44 - 1965.

[10] J.D.C. LITTLE, K.G. MURTY, D.W. SWEENEY, C. KARELL. An algorithm for the travelling salesman problem.
Operations research 11, N° 6 - 1963

[11] J.F. MUTH, G.L. THOMPSON - Industrial scheduling
Prentice Hall - 1963.

[12] M. NIVAULT
Thèse de spécialité, Toulouse - 1976.

[13] A.H.G. RINDOOY KAN - The machine scheduling problem.
Interfaculteit bedrijfskunde, Rotterdam - 1973.

[14] A.L. SAMUEL - Some studiesin machine learning using the game of checkers.
IBM J. res. dev. 3 - 1959
IBM J. res. dev. 11- 1967

[15] W.E. SMITH - Various optimizers for single-state production.
Nav. res. log. quart 3, N° 1 - 1956.

[16] M.B. WELLS - Elements of combinatorial computing.
Pergamon Press - 1971.