Objectif

L'objectif de cette thèse est de développer une approche particulière des algorithmes évolutionnaires qui auto-règle les paramètres de façon parallèle et contrôle dynamiquement leurs valeurs au fur et à mesure de l'évolution, relativement à la spécificité du problème.

Plan

Dans un premier point, on fournira une présentation générale des algorithmes évolutionnaires. On fournira également un bref résumé des points-clé et des différents domaines d'application des AEs.

Dans un second point, on exposera la problématique concernant le réglage/contrôle de paramètres dans les AEs. On présentera également les divers États-de-l'Art AEs proposés dans la littérature pour l'ajustement automatiquement des paramètres.

Dans un troisième point, on introduira deux approches préliminaires qui traitent le réglage dynamique des paramètres. La première change automatiquement la représentation des solutions d'une façon séquentielle. La seconde alterne la représentation d'une manière parallèle. Des applications à ces deux stratégies s'exprimeront sur différents problèmes d'optimisation difficiles.

Dans un quatrième point, on introduira notre contribution principale. C'est une variante parallèle des AEs appelée SEA (States based Evolutionary Algorithm en anglais). C'est une extension générale du cadre précédent dans le but du contrôle dynamique des paramètres.

Dans un cinquième point, une application du SEA sur le problème du sac à dos multidimensionnel (MKP) est faite. Des statistiques ont été également réalisées, ce qui permettra d'évaluer les performances de la méthode.

Dans un sixième point, on conclue avec des discussions générales et des perspectives.

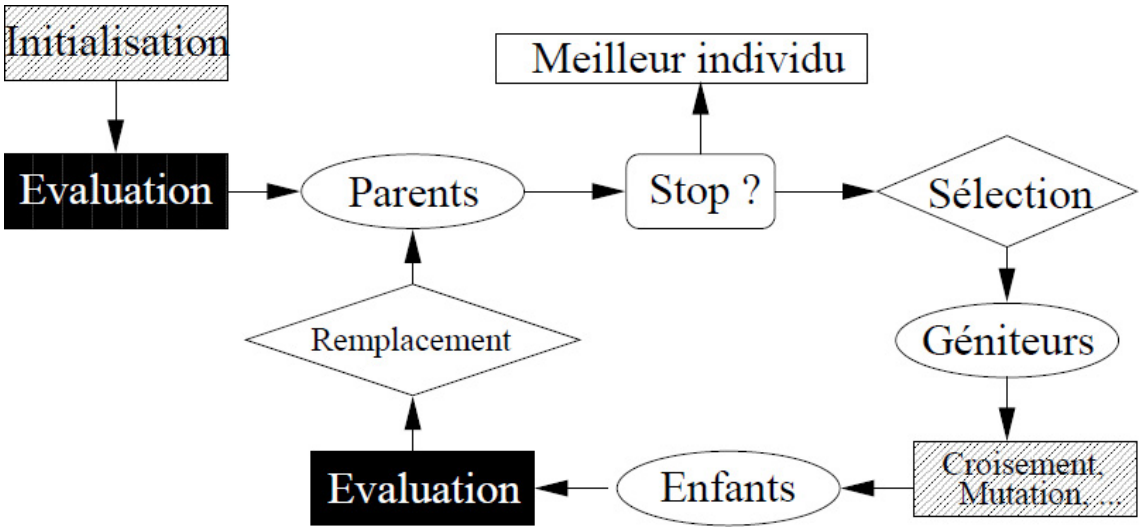
Algorithmes Évolutionnaires

Les Algorithmes évolutionnaires sont des méthodes d'optimisation stochastique basées sur une simulation brute de l'évolution naturelle des populations. Ils s'inspirent du principe de l'évolution des espèces décrit par Darwin.

Un AE est une méthode itérative qui utilise des opérateurs de variations stochastiques (croisement et mutation) sur un "pool" d'individu (la population). Chaque individu de la population représente une version encodée (solution) du problème.

En permettant aux seuls individus bien adaptés à l'environnement de se reproduire, la nature assure la pérennité de leurs meilleures caractéristiques pour former à chaque génération de nouveaux individus toujours mieux adaptés à leur environnement.

Schéma d'un AE



D'abord, on se dote d'un ensemble de solutions choisies au hasard : c'est la population de départ. À l'aide de la fonction d'adaptation, on évalue chacune d'entre elle. Pour commencer à construire la nouvelle génération, il faut deux parents choisis avec une probabilité d'autant plus grande que leur adaptation était bonne : c'est la phase de sélection. Partant de ces deux parents, on construit une descendance par croisement. Pour terminer, on procède au hasard à quelques mutations pour dynamiser l'exploration de l'espace des solutions et surtout d'éviter aux nouvelles générations de rester bloquées dans un optimum local. Il ne reste plus qu'à recommencer jusqu'à ce qu'on obtienne des solutions satisfaisantes au problème.

Points-clé des AEs

On cite ci-dessous deux points-clé des algorithmes évolutionnaires :

– La diversité génétique
Le réglage du mécanisme de sélection est décisif dans le comportement de l'algorithme évolutionnaire : un excès de sélection conduit parfois à une perte de diversité (et résulte en des zones inatteignables de l'espace de recherche), et une déficience peut mener à une marche aléatoire durant la recherche (donc pas de convergence).

– Le dilemme exploration/exploitation
A chaque étape de l'algorithme, il faut effectuer le compromis entre explorer l'espace de recherche, afin d'éviter de stagner dans un optimum local, et exploiter les meilleurs individus obtenus, afin d'atteindre de meilleurs valeurs dans le voisinage. Si les individus d'une population se ressemblent trop, la population risque de se résumer à l'évolution d'un seul individu dominant, réduisant ainsi l'exploration de l'espace de recherche. De même, un excès d'exploitation peut conduire à une convergence prématurée (enlisement dans un optimum local).

Applications des Algorithmes Évolutionnaires

Les AEs peuvent être utilisés pour résoudre une grande variété de problèmes d'optimisation globale et combinatoire tels que : les problèmes de télécommunication, de transport et de recherche opérationnelle.
Les applications des AEs sont également disponibles dans divers domaines tels que l'optimisation de forme en mécanique des solides.

Par exemple, pour résoudre le fameux problème du voyageur de commerce, on construit une population représentant un ensemble de parcours aléatoires. On sélectionne ensuite les meilleurs (les plus courts) que l'on croise entre eux pour obtenir une nouvelle population de parcours et ceci aussi longtemps qu'on le souhaite pour approcher le parcours optimum. Les résultats obtenus par des praticiens montrent que les AEs sont particulièrement adaptés à ce type de problème.

Par exemple, pour résoudre un problème de télécommunication ou de transport : il faut déterminer où installer 5 centres de distribution parmi 30 sites d'implantation possibles, de sorte que les coûts de transport entre ces centres et les clients soient minimum.

1 – Problématique et Présentation Générale *(Réglage/Contrôle des Paramètres)*

L'une des principales difficultés pour un utilisateur d'AEs consiste à choisir leurs paramètres de façon appropriée. Souvent, un paramètre ou valeur de paramètre peut opérer très bien au début de l'exécution d'un AE, mais se vérifier être inutile et trompeur au milieu ou à la fin de l'exécution.

Le réglage d'un AE pour une application donnée est une tâche difficile mais cruciale pour obtenir de bonnes performances. D'abord, il y a de nombreux paramètres à régler (codage, taille de la population, taux de mutation, …). Il y a également un grand nombre d'options et l'utilisateur aura seulement un peu de connaissance sur l'effet des paramètres choisis sur les performances de l'algorithme.

Pour ajuster les paramètres durant l'exécution, on a besoin des valeurs initiales de ces paramètres et des stratégies appropriées de contrôle. À leur tour, de telles alternatives peuvent être déterministes ou adaptatives. Les méthodes déterministes utilisent une simple équation (formule) pour calculer une valeur approximative de paramètre afin de l'ajuster selon le problème. Les méthodes adaptatives utilisent des informations sur l'état actuel de la recherche. Elles sont des méthodes efficaces lorsque l'unique sélection basée sur la fonction de fitness peut empêcher de mauvais paramètres de procéder largement à de futures générations.

État-de-l'Art AEs pour le Réglage/Contrôle des Paramètres

Dans le tableau suivant, on peut distinguer une variété des État-de-l'Art AEs proposés dans le but de réglage/contrôle des paramètres. Des méthodes déterministes dans un modèle parallèle/séquentiel ont été proposées pour l'échange/migration des individus ou le changement de représentation. Des méthodes adaptatives dans un modèle parallèle/séquentiel ont été également proposées pour l'ajustement de la taille de la population, du taux de mutation et bien d'autres paramètres.

Heuristique	**Concept**	**Type de Réglage de Paramètres**	**Modèle**
Mora et al. *A Parallel Evolutionary Algorithm Applied to the Minimum Interference Frequency Assignment Problem* (2006)	Déterministe	Échange/migration des individus	Parallèle
Spiessens et Manderick. *A Massively Parallel Genetic Algorithm* (1991)	Déterministe	Échange/migration des individus	Parallèle
Barbulescu et al. *Dynamic Representations and Escaping Local Optima: Improving Genetic Algorithms and Local Search* (2000)	Déterministe	Représentation	Séquentiel

Arabas et al. *GAVaPS - A Genetic Algorithm with Varying Population Size* (1994)	Adaptatif	Taille de population	Séquentiel
Eiben et al. *Evolutionary Algorithms with On-The-Fly Population Size Adjustment* (2004)	Adaptatif	Taille de population	Séquentiel
Lobo et al. *A Parameter-Less Genetic Algorithm* (1999)	Adaptatif	Taille de population	Parallèle
Smith et Fogarty. *Self Adaptation of Mutation Rates in a Steady State Genetic Algorithm* (1996)	Adaptatif	Taux de mutation	Séquentiel
Yang. *Adaptive Crossover in Genetic Algorithms Using Statistics Mechanism* (2002)	Adaptatif	Taux de croisement	Séquentiel
Mora et al. *A Parallel Evolutionary Algorithm Applied to the Minimum Interference Frequency Assignment Problem* (2006)	Déterministe	Échange/migration des individus	Parallèle
Spiessens et Manderick. *A Massively Parallel Genetic Algorithm* (1991)	Déterministe	Échange/migration des individus	Parallèle
Barbulescu et al. *Dynamic Representations and Escaping Local Optima: Improving Genetic Algorithms and Local Search* (2000)	Déterministe	Représentation	Séquentiel

Problématique / Solution

Donc la problématique se résume en le sujet du réglage/contrôle des paramètres, et la solution se résume en la proposition des variantes des AEs pour le changement dynamique des paramètres durant l'exécution.

2 – Approches Préliminaires

Le changement de codage a été largement abordé dans la littérature des AEs. Plusieurs problèmes d'optimisation peuvent être codés par une variété de représentations. Des codages et des opérateurs différents engendrent des espaces de recherche variés. La représentation doit être simple dans sa construction, doit minimiser l'épistasie (indépendance des gènes entre eux) et doit pouvoir coder toutes les solutions possibles d'un problème d'optimisation.

Donc, pour éviter un choix d'une représentation qui correspond pas au problème, des stratégies séquentielles/parallèles de changement dynamique de codage sont formulées, ce qui peut permettre une optimisation plus efficace.

Stratégies Séquentielles de Double Codage

Dans les propositions de double codage, le changement de représentation est considéré comme un réglage de paramètres préliminaire de manière séquentielle. En premier lieu, nous avons appliqué le concept de double codage en utilisant des techniques séquentielles telles que Periodic-EA, Aperiodic-EA, LocalOpt-EA, HomogPop-EA et SteadyGen-EA.

L'idée pour Periodic-EA est d'alterner entre deux codages lorsqu'un nombre donné de générations est atteint.

L'idée pour Aperiodic-EA est d'alterner entre deux codages lorsqu'un nombre aléatoire de générations est atteint.

LocalOpt-EA consiste à changer de codage lorsque le meilleur individu d'une population est un optimal local. L'idée était qu'un optimum local dans un codage n'est pas certainement un optimum local dans un autre codage. Cela permettra probablement d'échapper à des optima locaux et donc d'obtenir de meilleurs résultats.

L'idée pour HomogPop-EA est de changer la représentation lorsque la population devient homogène. Le critère d'homogénéité est mesuré par l'écart-type des fitness dans la population en comparaison avec un nombre réel donné ($\varepsilon \approx 0$).

Dans SteadyGen-EA, l'alternance est réalisée lorsque la valeur de la meilleure fitness n'est pas modifiée pour un nombre donné de générations. Ceci permet d'introduire une diversité génétique à la population.

Stratégie Parallèle de Double Codage

Dans une première phase, cette technique consiste à générer une population initiale pop de manière aléatoire. Puis, il convient de diviser cette population en deux sous-populations pop_1 et pop_2, ayant chacune une représentation distincte.

Puis, les deux sous-populations sont exécutées en parallèle avec deux AEs standard jusqu'à ce que la meilleure fitness de pop_1 ou pop_2 n'ait pas bougée pour un certain nombre de générations. Cette parallèle codification de solutions décrit l'apparition d'une proactivité sur deux niveaux.

Ensuite, pop_1 et pop_2 sont fusionnées en une seule population pop ayant le meilleur codage comme représentation. Le meilleur codage est sélectionné relativement à la sous-population ayant la moyenne de fitness la plus élevée. La fusion sera un moyen approprié pour assembler toutes les données développées dans une même population. À ce stade, les meilleurs individus répandus dans la population et les échanges réalisés par les opérateurs génétiques permettront de générer de meilleures structures.

Ensuite, un AE standard s'exécute avec pop jusqu'à ce que la meilleure fitness n'ait pas bougée pour un certain nombre de générations. Dans ce cas, on retourne à l'étape de division de pop. Le cycle Diviser-et-Fusionner continue jusqu'à ce qu'un nombre maximum de générations soit atteint.

Mise en Place des Expériences

Problèmes de Test

Nous avons retenu un total de cinq fonctions d'optimisation pour effectuer les expériences. Les fonctions choisies sont des problèmes de minimisation à valeurs réelles non contraintes et montrent différents degrés de complexité.

La première est Rosenbrock [F2] proposée par De Jong. Elle est unimodale (c.-à-d. elle ne contient qu'un seul optimum). [F2] a le minimum global au point (1, 1).

La seconde est Schaffer [F6] conçue par Schaffer. C'est un exemple de fonction multimodale (c.-à-d. contenant plusieurs optima locaux mais seulement un optimum global). [F6] a le minimum global à (0, 0).

La troisième, Rastrigin [F7], est un modèle typique d'une fonction hautement multimodale non linéaire. [F7] a le minimum global à (0, ..., 0).

La quatrième, Griewangk [F8], est aussi une fonction multimodale non linéaire. Elle a une complexité élevée et a le minimum global à (0, 0).

La cinquième, Schwefel [F9], est également une fonction multimodale non linéaire. Elle plus facile que [F7] et est caractérisée par un second meilleur minimum qui est éloigné de l'optimum global. [F9] a le minimum global à (420, 9687, ..., 420, 9687).

Nom	Expression	Intervalle	Dimension
$[F2]$	$f_2(x_i) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$	$[-2.048 : 2.048]$	2
$[F6]$	$f_6(x_i) = 0.5 + \frac{\sin^2(\sqrt{x^2+y^2})-0.5}{(1+0.001(x^2+y^2))^2}$	$[-100 : 100]$	2
$[F7]$	$f_7(x_i) = 200 + \sum_{i=1}^{20}(x_i^2 - 10\cos(2\pi x_i))$	$[-5.12 : 5.12]$	20
$[F8]$	$f_8(x_i) = 1 + \sum_{i=1}^{10}(\frac{x_i^2}{4000}) - \prod_{i=1}^{10}(\cos(\frac{x_i}{\sqrt{i}}))$	$[-600 : 600]$	10
$[F9]$	$f_9(x_i) = 10V + \sum_{i=1}^{10}(-x_i \sin(\sqrt{\lvert x_i \rvert}))$	$[-500 : 500]$	10

Description des Tests :

Dans toutes les expérimentations faites, le codage binaire standard et un codage de Gray sont utilisés dans les stratégies de double codage.
Lors du calcul des valeurs réelles des solutions pour chacune des méthodes proposées, la valeur de fitness est égale à la valeur de la fonction calculée selon l'expression correspondante fournie dans le Tableau précédent.

- Nombres réels sont calculés comme suit :

$$x = a + \frac{b-a}{2^N - 1}\sum_{i=0}^{N-1} x_i 2^i$$

  - x est la valeur réelle de la solution codée en binaire standard
  - N est la taille de la chaîne binaire
  - a et b sont respectivement les bornes inférieures et supérieures de l'intervalle de recherche

- Formules suivantes sont utilisées pour passer du binaire standard $(s_{k-1}...s_1s_0)$ en Gray $(g_{k-1}...g_1g_0)$:

$$g_i = s_{i+1} \oplus s_i \quad \text{et} \quad s_i = s_{i+1} \oplus g_i$$

Réglage des Paramètres

Dans les propositions déjà définies, un AE simple est utilisé avec les valeurs standards de paramètre :
– Taille de population = 100
– Mécanisme de sélection : Sélection de Tournoi avec une taille de tournoi = 2.
– Mécanisme de croisement : Croisement à 1-Point avec un taux = 0.6.
– Mécanisme de mutation : Mutation par Bit avec un taux = 1.0.
– Modèle de remplacement : Remplacement Générationnel avec Élitisme.
– Critères de d'arrêt d'algorithme : les exécutions s'arrêtent lorsqu'un nombre maximum de générations (= 3500) est atteint.

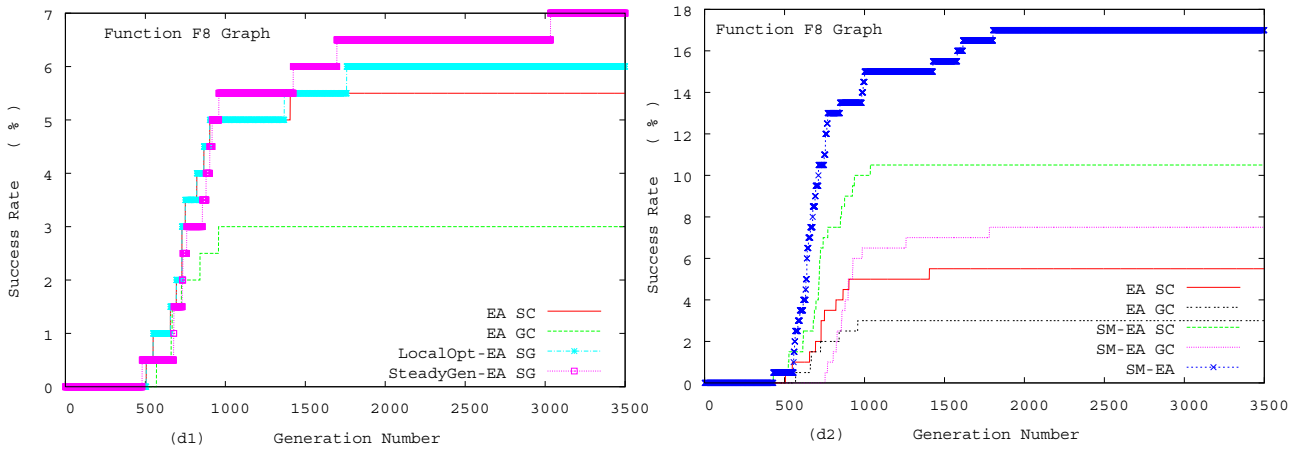
Résultats Expérimentaux

Les stratégies de double codage sont testées sur les fonctions objectives. Une comparaison est faite avec un AE simple. Le Tableau suivant présente les résultats expérimentaux obtenus lors de 200 exécutions et restitués à la dernière génération. Ce tableau montre le nombre de générations à l'optimum (GNTO) et le taux de succès (SR), tout en indiquant le score le plus élevé en gras.

La valeur GNTO correspond au nombre de générations nécessaires pour atteindre l'optimum global en moyenne sur toutes les exécutions.
La valeur SR représente le pourcentage du nombre de fois où la solution optimale est trouvée à la fin de toutes les exécutions.

EA signifie un algorithme évolutionnaire simple, S pour une exécution avec le codage binaire standard, G pour une exécution avec le codage de Gray, SG signifie que le codage binaire standard était le codage de départ, et GS lorsque le codage de Gray était le codage de départ.

Proposition	[F2]		[F6]		[F7]		[F8]		[F9]	
	GNTO	SR%	GNTO	SR%	GNTO	SR%	GNTO	SR%	GNTO	SR%
EA_{SC}	3500+	31	3500+	37	3500+	0	3500+	6	3500+	0
$SM\text{-}EA_{SC}$	3500+	30	3500+	48	3500+	2	3500+	11	3500+	0
EA_{GC}	3500+	99	3500+	43	2957	99	3500+	3	2395	94
$SM\text{-}EA_{GC}$	3256	99	3500+	52	3362	98	3500+	8	2413	97
$SM\text{-}EA$	**3139**	**100**	**3500+**	**57**	**2940**	**100**	**3500+**	**17**	**2025**	**100**
$Periodic\text{-}EA_{SG}$	3500+	86	3500+	51	**3191**	**98**	3500+	5	3480	79
$Periodic\text{-}EA_{GS}$	3500+	91	3500+	46	3500+	80	3500+	4	**2279**	**92**
$Aperiodic\text{-}EA_{SG}$	3500+	90	3500+	51	3500+	93	3500+	3	3009	89
$Aperiodic\text{-}EA_{GS}$	3500+	92	**3500+**	**52**	3230	93	3500+	3	2818	90
$LocalOpt\text{-}EA_{SG}$	3500+	76	3500+	49	3500+	88	3500+	6	2480	93
$LocalOpt\text{-}EA_{GS}$	3500+	80	3500+	48	3491	95	3500+	5	2480	94
$HomogPop\text{-}EA_{SG}$	3500+	31	3500+	41	3500+	0	3500+	2	3500+	0
$HomogPop\text{-}EA_{GS}$	**3500+**	**99**	3500+	38	3001	95	3500+	3	2395	94
$SteadyGen\text{-}EA_{SG}$	3500+	87	3500+	46	3381	95	**3500+**	**7**	2453	86
$SteadyGen\text{-}EA_{GS}$	3500+	88	3500+	51	3254	98	3500+	4	2894	82



Représentation de l'évolution du taux de succès au cours des générations pour la fonction [F8] :

- comparaison entre les stratégies séquentielles et un AE simple dans (a)
- comparaison entre la stratégie parallèle et un AE simple dans (b)

<u>Interprétations des Résultats</u>

Pour les stratégies séquentielles, les résultats fournis montrent que chacune de ces propositions a pu légèrement améliorer la performance d'un AE simple pour un problème donné. Nous pouvons dire qu'elles ont produit des résultats qui sont aussi bons que ceux d'exécution d'un AE simple avec une représentation unique. Cela est du au fait qu'un optimum local dans un codage n'est pas nécessairement un optimum local dans un autre codage.

Pour la stratégie parallèle, on peut dire que SM-EA a produit des résultats relativement élevés par rapport aux autres algorithmes. Les statistiques et les représentations graphiques montrent, pour chaque fonction objective, une vitesse de convergence assez importante pour SM-EA en comparaison avec d'autres méthodes. Cela suggère que, tout en étant possible de contrôler avec précision les paramètres d'un AE, une très bonne performance peut être obtenue.

Par ailleurs, SM-EA montre son efficacité envers SM-EA-S et SM-EA-G, ce qui prouve que ce n'est pas uniquement l'effet de parallélisme qui a fait la méthode évoluer au cours des générations, mais aussi le fait d'intégrer différentes représentations a contribué à une meilleure performance et à une vitesse de convergence remarquable.

Synthèse sur les Approches Préliminaires

Les stratégies de double codage provoquent une interaction entre différentes représentations, ce qui aide à transformer la représentation du paramètre du problème. Elles permettent aussi à éviter de se bloquer dans des optima locaux. Cela est faisable parce qu'un optimum local sous une représentation n'est pas forcément un optimum local sous l'autre.

La stratégie parallèle de changement dynamique de représentation a fourni de meilleurs résultats que les autres méthodes. Ceci nous a incité a continué l'étude dans cette direction tout en généralisant cette approche afin qu'elle puisse être applicable à n'importe quel paramètre d'un AE. Dans la suite, on verra comment attribuer un état à chaque paramètre/valeur de paramètre d'un AE dans le but d'adapter un tel paramètre à la spécification du problème.

3 – Contribution Principale

Maintenant, on introduit la notion d'état dans les algorithmes évolutionnaires. Un état peut symboliser n'importe quel paramètre ou valeur de paramètre de l'algorithme. De cette façon, on peut imaginer une évolution au niveau des états de l'algorithme et des paramètres à ajuster.

Notre contribution principale est une variante parallèle des AEs appelée algorithme évolutionnaire à états (SEA pour States based Evolutionary Algorithm). Elle est basée sur l'idée d'assembler plusieurs états simultanément dans le but d'ajuster les paramètres en ligne. De cette manière, l'échange de données à travers l'espace de solutions aidera à faire mieux évoluer la recherche.

Algorithme Évolutionnaire à États (SEA)

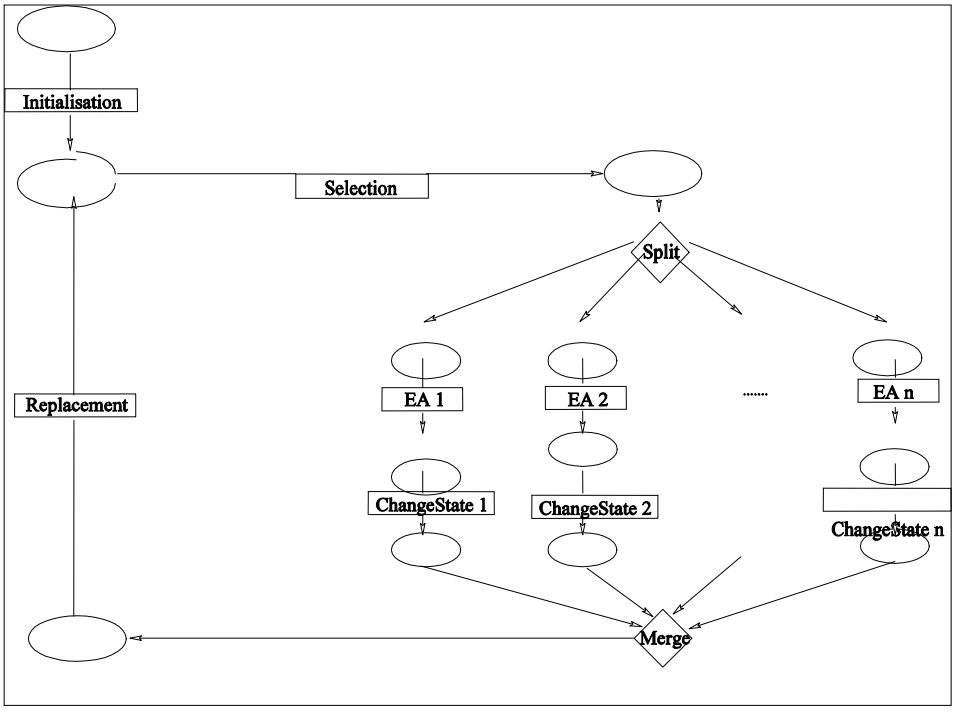
À chaque solution de la population d'un SEA est affecté un état particulier représenté par un nombre entier entre [1, n] où n est le nombre des états du SEA. Le SEA exécute un algorithme évolutionnaire simple EA_i (i = 1, ..., n) sur la sous-population de solutions ayant le même état i.

Un bon état dans le SEA correspond à l'état le plus adéquat à la recherche pour un problème donné. C'est un état qui s'accorde bien à la structure de la fitness et qui rend l'optimisation plus efficace. Le principe du SEA est de "choisir" le bon état tout en gardant les autres états dans le "pool". Le SEA pourrait ainsi ajuster les tailles des sous-populations des EA_i (i = 1, ..., n) afin d'avoir une distribution optimale du nombre de solutions dans chaque état.

L'exploitation d'un état dans un SEA se fait alors par l'affectation d'une population de taille assez importante à cet état. L'exploration de l'ensemble des états dans un SEA s'effectue par l'intermédiaire de l'opérateur de changement d'état qui change l'état des solutions aléatoirement. Cela permettra de conserver un nombre minimum de solutions dans chaque état afin de maintenir une diversité de paramètres/valeurs de paramètres. De cette manière, un compromis entre l'exploitation des "meilleurs" états et l'exploration des autres états est réalisé dans le SEA.

Une itération de l'algorithme est la succession des opérateurs stochastiques : Selection (sélection), Split (division), EA_i (AE simple), ChangeState (changement d'état), Merge (fusion) et Replacement (remplacement).

L'opérateur Selection crée une population de solutions, favorisant celles avec de meilleures fitness sans aucune considération de leurs états.
L'opérateur Split divise la population en n sous-populations suivant les états des solutions. Chaque sous-population sera composée de toutes les solutions ayant le même état.
L'opérateur EA_i est un algorithme évolutionnaire simple correspondant à l'ensemble des solutions qui existent dans l'état d'indice i.
ChangeState est un opérateur qui est appliqué uniformément à chaque sous-population, modifiant au hasard l'état des solutions.
L'opérateur Merge fusionne les n sous-populations dans une population unique afin de procéder à l'étape de remplacement des solutions.
L'opérateur Replacement est un opérateur de remplacement générationnel standard. Il crée une nouvelle population pour la prochaine itération à partir des sous-populations de tous les EA_i (i = 1, ..., n) et de l'ancienne population de solutions.



Caractéristiques du SEA

– Le SEA est une méthode parallèle dont l'exécution sur des machines différentes peut réduire le coût de calcul (distribution des tâches sur plusieurs processeurs).

– Il est basé sur un ensemble d'heuristiques de bas niveau faciles à implémenter qui sont responsables à mener l'algorithme dans un système adaptatif.

– Il conserve une variation génétique importante durant la recherche grâce à l'intégration d'une diversité de paramètres/valeurs de paramètres.

– Il est plus puissant qu'un AE standard en raison d'une recherche séparée dans plusieurs régions de l'espace de solution.

– Il adapte le bon paramètre/valeur de paramètre au problème en lui attribuant une taille de population plus large que celle des autres paramètres/valeurs de paramètres.

– Il ajuste dynamiquement les tailles des sous-populations et maintient conséquemment un compromis entre l'exploration/exploitation de l'espace des états.

Opérateur de Changement d'État Dirigé par la Topologie d'´État dans le SEA

Dans la description du SEA, l'opérateur de changement d'état est appliqué uniformément à tous les états. Dans cette section, nous proposons une nouvelle stratégie pour l'opérateur de changement d'état dans le SEA. Cette technique peut être appliquée lorsque c'est possible d'avoir une notion de voisinage entre les états.

Dans cette topologie d'état, la probabilité de passer d'un état aux états voisins est plus grande que de passer aux autres états. Elle permet d'explorer l'espace des états. Ceci peut contribuer davantage à l'exploration de l'espace de recherche, en supposant que de bonnes solutions seront propagées de manière homogène dans la topologie d'états sous-jacente.

4 – Étude Expérimentale : Auto-adaptation du Taux de Mutation dans le SEA pour résoudre le problème du sac à dos multidimensionnel (MKP)

L'application d'un taux de mutation incorrect à des individus de la population d'un AE peut mener à une convergence trompeuse vers des solutions de qualité médiocre. Par conséquent, le réglage du taux de mutation est crucial dans le processus des AEs. Ceci est particulièrement vrai pour le problème du sac à dos multidimensionnel (MKP).

Plusieurs heuristiques ont été appliquées au problème MKP avec grand succès. Le SEA, considéré comme un cadre automatisé pour le contrôle des paramètres, peut s'adapter en fonction des contraintes spécifiées dans ce problème.

Pour valider le SEA et montrer son utilité et les avantages qu'il peut apporter à la recherche, nous avons choisi de tester cette approche sur un problème d'optimisation combinatoire NP difficile tel que le MKP. Des expériences ont été menées sur le MKP, où les états du SEA correspondent aux différents taux de mutation par Bit.

Problème du Sac à Dos Multidimensionnel à Variable 0 – 1

Le problème du sac à dos multidimensionnel, MKP, est un problème d'optimisation combinatoire bien étudié. Le MKP est fortement NP difficile se produisant dans un grand nombre d'applications différentes. Il peut être défini par les énoncés mathématiques suivants :

$$maximiser \quad z = \sum_{j=1}^{n} p_j x_j \tag{1}$$

$$sujet \quad de \quad \sum_{j=1}^{n} w_{ij} x_j \leq c_i, \quad i = 1, \ldots, m, \tag{2}$$

$$où \quad x_j \in \{0, 1\}, \quad j = 1, \ldots, n. \tag{3}$$

Plus précisément, ils existent n articles avec des profits $p_j > 0$ et m ressources avec des capacités $c_i > 0$. Chaque article j consomme une quantité $w_{ij} \geq 0$ de chaque ressource i. L'objectif, selon l'Équation 1, est de sélectionner un sous-ensemble d'articles avec un profit total maximum; toutefois, les articlelusieurs heuristiques et techniques exactes pour résoudre des instances de tailles différentes du MKP ont été proposées dans la littérature.

Parmi ces méthodes, on peut citer une variante efficace des algorithmes génétiques proposée par Chu et Beasley en 1998. Dans cette technique, uniquement les solutions faisables étaient prises en compte durant la recherche. Une heuristique a été utilisée pour convertir les solutions infaisables en des solutions faisables.

La Recherche à Voisinage Variable (VNS) a été utilisée pour résoudre ce problème. Une nouvelle variante du VNS a été proposée par Puchinger et Raidl en 2007. Dans leur algorithme, l'ordre de visite des voisinages est déterminé dynamiquement en résolvant leurs relaxations.

À notre connaissance, les deux méthodes produisant actuellement les meilleurs résultats sur certaines grandes instances du MKP, ont été proposées par Vasquez et Vimont (V&V) en 2005 et par Wilbaut et Hanafi (W&H) en 2009 :

- Dans V&V, il s'agit d'une approche hybride basée sur la recherche tabu. L'espace de recherche est réduit et compartimenté par des contraintes additionnelles, fixant ainsi le nombre total d'articles à emballer. Les limites de ces contraintes sont calculées en résolvant une relaxation à l'aide de la programmation linéaire.

- Dans W&H, il s'agit d'une méthode utilisant plusieurs algorithmes de recherche locale pour résoudre une série de petits sous problèmes du MKP. Ensuite, des heuristiques basées sur la programmation linéaire sont utilisées afin de trouver une solution optimale du problème.

SEA-de-mutation pour Résoudre le MKP

Le réglage du taux de mutation par Bit a une grande influence sur les performances de l'algorithme. Lorsqu'il est trop large, l'algorithme effectue une marche aléatoire sur l'espace de recherche, il y a trop d'exploration. Lorsqu'il est trop petit, l'algorithme reste coincé dans des optima locaux, puisqu'il y a trop d'exploitation.

Le SEA est une approche parallèle capable d'utiliser plusieurs taux de mutation en favorisant une valeur différente pour chaque période. Le changement consécutif des taux de mutation sert à introduire assez de variations dans l'algorithme. Il permet de maintenir une diversité assez importante dans les structures de la population. Afin d'exploiter cette stratégie, nous développons une variante du SEA où un état représente un AE simple avec un taux de mutation différent. Cette version est appelée SEA-de-mutation (m-SEA).

SEA-de-mutation Topologique pour Résoudre le MKP

Puisque c'est possible d'avoir une notion de voisinage entre les états du m-SEA (déjà introduit), on essaiera dans cette section d'exploiter la topologie d'états présentée précédemment. Les structures de voisinage dans le m-SEA peuvent être obtenues en calculant les distances entre les différents états puisqu'ils sont représentés par des nombres entiers.

De cette manière, les transitions d'état des individus seront effectuées/favorisées entre états. Cette version du m-SEA, où l'opérateur de changement d'état est dirigé par la topologie d'états, est appelée m-SEA Topologique (m-T-SEA).

Configuration des Expériences

Les tests ont été réalisés sur un ensemble de neuf parmi les plus grandes instances du MKP avec n = 500 articles, $m \in \{5, 10, 30\}$ contraintes, et taux de tension $\alpha \in \{0.25, 0.5, 0.75\}$ (nous utilisons la bibliothèque de référence de Chu et Beasley pour nos tests disponible sur : http://people.brunel.ac.uk/~mastjjb/jeb/info.html).

Dans les expérimentations effectuées :

– Le MKP, étant à maximiser avec des variables 0–1, est codé en chaînes binaires de taille n = 500 bits.
– 5 états sont utilisés dans m-SEA et m-T-SEA.
– Un algorithme évolutionnaire simple EA_i (i=1, …, 5) avec les composants standards recommandés par Goldberg (1989) est utilisé dans chaque état.
– La mutation classique de par Bit est utilisée avec un taux de mutation = 1.0.
– Ainsi, chaque état représente un EA_i (i=1, …, 5) avec un taux de mutation par Bit = i / 500.
– La taille de la population de m-SEA et m-T-SEA est égale à 1000.
– La sélection de tournoi standard est utilisée avec une taille de tournoi = 10.
– Le remplacement est "générationnel" avec "élitisme".
– Le croisement à un point est utilisé avec un taux de croisement = 0.6.
– Le changement d'état est effectué sur des solutions arbitraires avec un taux fixe = 0.1.
– Chacun des algorithmes s'arrête lorsqu'un nombre maximal de générations = 10000 est abouti.

Expérimentations et Résultats

Nous avons utilisé la Bibliothèque EO (disponible sur : http://eodev.sourceforge.net) pour coder les algorithmes testés et les fonctions objectives dans un code conforme à la norme ANSI-C++.

Pour évaluer les résultats de notre contribution, 30 exécutions indépendantes ont été effectuées sur les instances du MKP. Une comparaison est faite avec des AEs simples pour l'ensemble des taux de mutation utilisés. Une comparaison est également faite avec une instance du SEA (S-SEA) où l'opérateur de changement d'état

n'intervient pas dans le processus (S-SEA a été développé pour tester l'effet du parallélisme sur le fonctionnement du SEA).

Les résultats de test sont enregistrés pour les solutions finales (à la dernière génération) et moyennés sur les 30 exécutions. Précisément, on a calculé :

– la valeur de la meilleure fitness (m−fit).
– Le pourcentage de gap (%−gap). Formellement, pour une solution avec une valeur de fitness z, le %−gap est égal à $(z^{pk} - z)/z^{pk}$. 100%, où z^{pk} représente la valeur optimale de l'instance précédemment obtenue par V&V ou W&H.

Performance des Algorithmes Évolutionnaires Simples sur le MKP

Ce tableau fournit les résultats des AEs simples pour l'ensemble des taux de mutation utilisés. On voit très bien que le premier AE avec un taux de mutation le plus faible était le plus adéquat pour la recherche sur le MKP.

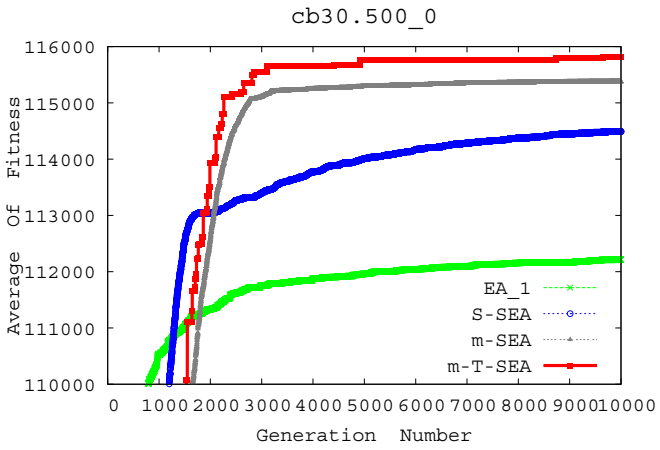
Instance	EA_1		EA_2		EA_3		EA_4		EA_5	
	$m-fit$	$\%-gap$	$m-fit$	$\%-gap$	$m-fit$	$\%-gap$	$m-fit$	$\%-gap$	$m-fit$	$\%-gap$
$cb5.500\_4$	119763	2.08962	116186	5.01394	112341	8.15736	110961	9.28556	110738	9.46787
$cb5.500\_17$	216020	1.00589	212062	2.8197	209610	3.94336	208591	4.41033	205262	5.93589
$cb5.500\_24$	299069	0.42385	296153	1.39474	293984	2.11692	292960	2.45786	291068	3.08781
$cb10.500\_5$	116874	2.20076	113077	5.37806	110566	7.47925	107889	9.71934	106776	10.6507
$cb10.500\_18$	211858	1.17734	207663	3.13413	205198	4.28394	203633	5.01395	201826	5.85683
$cb10.500\_27$	294807	0.563617	291682	1.61766	289933	2.20758	289386	2.39208	287954	2.87509
$cb30.500\_0$	113174	2.48328	108505	6.50634	105604	9.006	104778	9.71772	103031	11.223
$cb30.500\_12$	213446	1.17234	209163	3.15541	206255	4.50185	203655	5.70567	203576	5.74225
$cb30.500\_25$	295069	0.654182	292450	1.53596	290226	2.28476	288386	2.90426	287704	3.13388

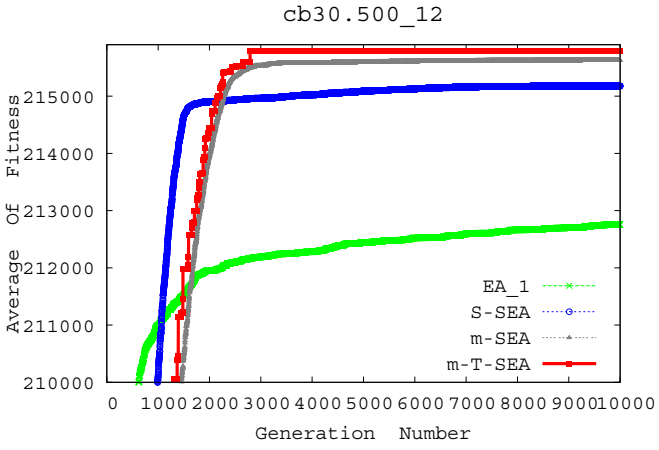
Performance des Variantes du SEA sur le MKP

Ce tableau fournit les résultats de S-SEA, m-SEA et m-T-SEA en comparaison avec les méthodes les plus récentes fournissant les meilleures solutions pour le MKP. On voit très bien la performance la plus élevée de m-T-SEA vis-à-vis les autres instances du SEA.

Instance	z^{pk}	*S-SEA*		*m-SEA*		*m-T-SEA*	
	$m-fit$	$m-fit$	$\%-gap$	$m-fit$	$\%-gap$	$m-fit$	$\%-gap$
*cb*5.500_4	122319	122079	0.196208	122272	0.03842	122319	0.00
*cb*5.500_17	218215	218066	0.0682813	218163	0.02383	218166	0.02245
*cb*5.500_24	300342	300225	0.0389556	300296	0.01532	300342	0.00
*cb*10.500_5	119504	119054	0.376556	119314	0.15899	119377	0.10627
*cb*10.500_18	214382	214042	0.158595	214252	0.06064	214265	0.05458
*cb*10.500_27	296478	296217	0.0880335	296392	0.02901	296442	0.01214
*cb*30.500_0	116056	115050	0.866823	115706	0.30158	115817	0.20594
*cb*30.500_12	215978	215431	0.253267	215780	0.09168	215793	0.08566
*cb*30.500_25	297012	296614	0.134001	296883	0.04343	296892	0.04040

Représentations Graphiques des Performances (Moyenne de Fitness)

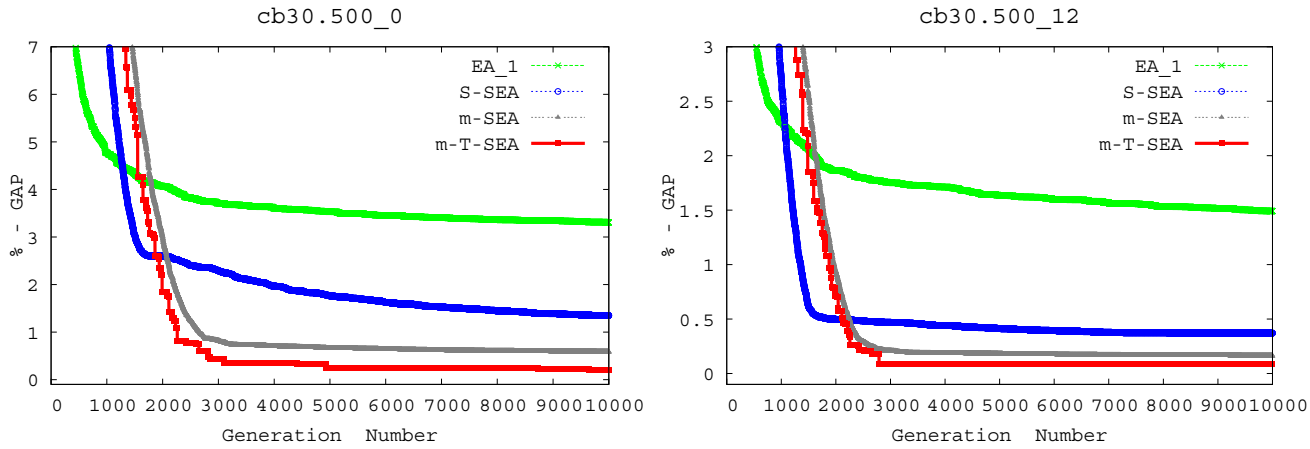




Deux instances du MKP illustrant les performances moyennes de EA_1, S-SEA, m-SEA et m-T-SEA :

- L'axe des x représente l'indice de génération.
- L'axe des y représente la meilleure fitness moyennée sur les 30 exécutions.

Représentations Graphiques des Performances (%-gap)



Deux instances du MKP illustrant les performances moyennes de EA_1, S-SEA, m-SEA et m-T-SEA :

- Axe des x représente l'indice de génération.
- Axe des y représente le %-gap des solutions moyenné sur les 30 exécutions.

Interprétations des Résultats

Bien que notre objectif principal ne soit pas de concurrencer les algorithmes d'État-de-l'Art pour l'ensemble des instances complexes du MKP, nous avons obtenu de très bonnes solutions (solutions concurrentes) pour plusieurs instances, et des résultats relativement bons qui en général ne sont pas trop éloignés des meilleures solutions connues pour les autres instances.

On peut remarquer aussi que m-SEA surpasse distinctement S-SEA pour toutes les instances du MKP. Cela signifie que dans m-SEA, des paramètres additionnels contrôlent les tailles des sous-populations et la migration des individus. Donc, le SEA intègre, en plus de son architecture hautement parallèle, différents niveaux d'évolution qui contribuent à l'obtention de bonnes performances.

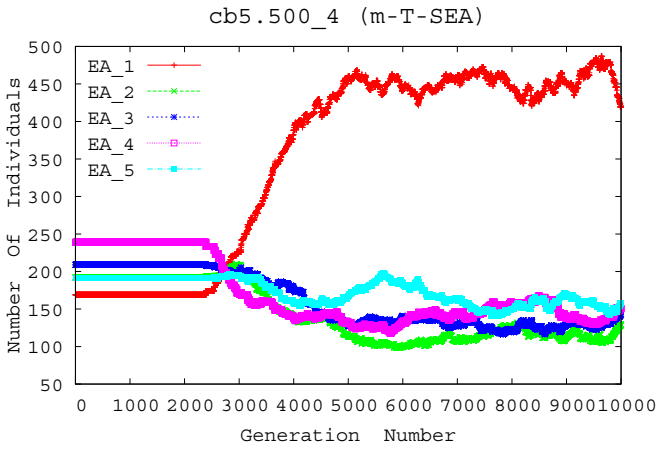
De l'autre côté, les résultats obtenus par m-T-SEA sont beaucoup plus meilleurs que les autres instances du SEA. Cela montre que l'opérateur de changement d'état entraîné par la topologie des états influence positivement les performances du SEA. Ce choix permet de garantir une synchronisation implicite entre états voisins. Ceci est basé sur le fait que les changements introduits par la mutation peuvent ne pas avoir un tel effet disruptif sur le chromosome et ensuite, de meilleures performances peuvent être obtenues.

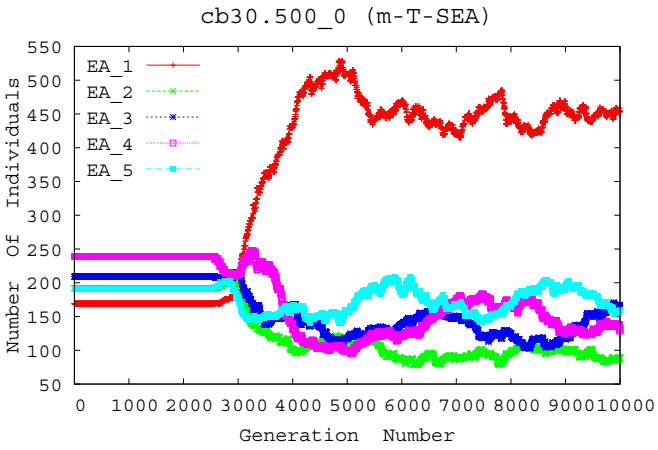
Dynamique de l'Évolution des États dans le SEA

La sélection de l'état approprié dans le SEA est effectuée en fonction des valeurs de fitness des solutions actuelles. Si les solutions d'un état donné sont meilleures que celles des autres états, alors l'opérateur de sélection choisit plus souvent ces solutions. Par conséquent, la taille de la sous-population de l'état correspondant augmente après une itération du SEA lorsque la valeur de la meilleure fitness a été modifiée.

Afin d'observer la dynamique inter-état, nous avons tracé le nombre total d'individus dans chaque sous-population à travers toutes les itérations (toujours en se basant sur les expériences décrites précédemment). Les résultats enregistrés sont moyennés sur les 30 exécutions.

Représentations Graphiques de la Dynamique des États dans le SEA





Deux instances du MKP illustrant la dynamique des états dans m-T-SEA :

- L'axe des x représente l'indice de génération.
- L'axe des y représente l'évolution du nombre d'individus dans chaque état.

Interprétation de la Dynamique des États dans le SEA

Dans les figures précédentes, on voit distinctement que l'état 1 dans m-T-SEA a submergé tous les autres états en ayant le plus grand pourcentage d'individus. Cela est conforme avec les résultats obtenus sur les performances des AEs simples sur le MKP qui montrent que EA_1 était l'AE simple le plus adéquat à la recherche sur le MKP. Ces représentations numériques justifie bien que le SEA est capable de "favoriser" le "meilleur" ´état lorsque c'est possible tout en conservant les autres états dans le système.

Synthèse sur le SEA

Le SEA est capable d'adapter les paramètres de l'algorithme selon le problème objectif. Il permet également d'ajuster les tailles des sous-populations en fonction des performances de leurs états correspondants.

Une lecture profonde des résultats obtenus nous permet de dire que le SEA n'est pas un méta-optimiseur des états : même s'il ne "sélectionne" pas uniquement le "meilleur" état, il bénéficie certainement de l'opération quasi-parallèle des AEs simples possédant une diversité de valeurs paramètres. Si au début, au milieu ou à la fin d'une exécution, une valeur de paramètre n'est pas parfaitement appropriée pour une instance donnée du problème, alors une autre valeur peut contribuer à produire de bons résultats et par conséquent à améliorer la performance de l'algorithme.

Conclusion et Perspectives

L'idée principale dans cette thèse était de contribuer au vaste domaine du contrôle des paramètres de l'algorithme suivant le problème de test.

Nous avons commencé notre étude par des propositions séquentielles et parallèles de double codage pour adapter la représentation des solutions à la fitness du problème.

Ensuite, on a introduit la notion d'états et du meilleur état aux AEs. Un état peut représenter n'importe quel paramètre/valeur de paramètre de l'algorithme.

Toujours en suivant l'idée de rester au plus proche de l'AE standard, nous avons proposé une implémentation parallèle des algorithmes évolutionnaires où chaque AE synchrone représente un état différent de l'espace de recherche.

L'algorithme ´évolutionnaire à état (SEA) que nous avons proposé semble être une approche utile pour contrôler les paramètres de l'algorithme en ligne. Le SEA, suggère une amélioration des performances des AEs grâce à des ajustements automatiques des paramètres suivant la structure du problème à optimiser.

Le SEA intègre une diversité de paramètres de la sorte qu'il introduise en même temps une diversité génétique importante à la population et à la recherche de nouvelles structures. Il serait intéressant dans des travaux ultérieurs d'étendre ses résultats à une famille de problèmes d'optimisation combinatoire et réelle.

Notre thèse n'était pas question de se comparer directement aux État-de-l'Art AEs pour des problèmes d'optimisation spécifiques, l'idée étant de proposer un algorithme évolutionnaire efficace dans le cadre d'une utilisation simple sans réglage ni spécialisation.

Nous pouvons envisager deux principales directions pour de futurs travaux, l'une portant sur l'étude des propriétés des paramètres et leurs rapports aux algorithmes d'optimisation, et l'autre sur la conception de nouvelles techniques d'ajustement automatique de ces paramètres.

Une étude ultérieure suggère aussi que d'autres implémentations du SEA peuvent toujours être améliorées en réduisant les paramètres définis par l'utilisateur et en les rendant automatiquement ajustables en fonction des mesures extraites du processus. Ce concept peut rendre le SEA une approche réelle visant à ajouter un progrès important dans la direction des algorithmes évolutionnaires les plus performants.