RAIRO. RECHERCHE OPÉRATIONNELLE

CHARLES FLEURENT
JACQUES A. FERLAND

Algorithmes génétiques hybrides pour l'optimisation combinatoire

RAIRO. Recherche opérationnelle, tome 30, n° 4 (1996), p. 373-398

<http://www.numdam.org/item?id=RO_1996__30_4_373_0>

Recherche opérationnelle/Operations Research
(vol. 30, n° 4, 1996, pp. 373-398)

ALGORITHMES GÉNÉTIQUES HYBRIDES POUR L'OPTIMISATION COMBINATOIRE (*)

par Charles FLEURENT et Jacques A. FERLAND ([1])

Résumé. – *Depuis quelques années, des algorithmes génétiques ont été adaptés pour traiter certains problèmes d'optimisation combinatoire. Dans la plupart des cas cependant, les algorithmes génétiques purs sont moins performants que d'autres méthodes heuristiques. Nous examinons certaines approches hybrides récentes où l'utilisation d'opérateurs génétiques améliore la performance de méthodes heuristiques déjà existantes.*

Mots clés : Algorithmes génétiques, optimisation combinatoire, méthodes hybrides.

Abstract. – *For many years, genetic algorithms have been adapted to solve various combinatorial optimization problems. However, in most cases, pure genetic algorithms are outperformed by other heuristic methods. In this paper, we examine some recent hybrid approaches in which the use of genetic operators improves the performance of existing heuristic procedures.*

Keywords: Genetic algorithms, combinatorial optimization, hybrid methods.

1. INTRODUCTION

On peut modéliser un problème d'optimisation combinatoire comme suit :

$$\begin{array}{l} \text{Min } f(x) \\ \text{s.à } x \in \mathcal{S} \end{array} \qquad \text{(P)}$$

où $\mathcal{S}$ est un ensemble fini. Bien que l'on connaisse des algorithmes efficaces pour certains de ces problèmes (e.g. programmation linéaire), d'autres par contre sont beaucoup plus difficiles. Parmi ces problèmes, notons par exemple ceux du commis-voyageur [39], de la coloration de graphe [50],

(*) Reçu en juin 1994.
([1]) Université de Montréal, Département d'Informatique et de Recherche Opérationnelle, 2900 Édouard-Montpetit, C.P. 6128 Succ. Centre-Ville, Montréal, P.Q. Canada H3C 3J7.

Recherche opérationnelle/Operations Research, 0399-0559/96/04/$ 7.00

ou de l'affectation quadratique [52]. Pour résoudre ces problèmes, certaines méthodes exactes sont disponibles [35, 36], mais ne sont utiles en pratique que lorsque la taille des problèmes est relativement petite [21].

Pour les problèmes de grande taille, on doit faire appel à des méthodes heuristiques. Parmi les approches les plus populaires, on compte actuellement les méthodes de recherche locale [43], de recherche avec tabous [24], et de recuit simulé [38]. Les algorithmes génétiques [33] constituent une autre famille de méthodes heuristiques pouvant être utilisées pour ces problèmes. Dans ce travail, nous faisons un survol de récentes adaptations de ces méthodes pour résoudre différents problèmes d'optimisation combinatoire. Dans la plupart des cas, il faut constater que les algorithmes génétiques purs sont moins performants que d'autres méthodes heuristiques. Nous présentons toutefois certaines approches récentes où la combinaison d'opérateurs génétiques et de méthodes déjà existantes donne lieu à des algorithmes heuristiques parmi les plus puissants actuellement disponibles.

Dans la prochaine section, les éléments de base des algorithmes génétiques sont rappelés. Par la suite, certains problèmes particuliers sont considérés, pour lesquels différentes représentations de leurs solutions et différents opérateurs génétiques sont définis. La quatrième section s'intéresse à la mesure de la diversité des populations, et aux actions possibles pour la maintenir. Dans la cinquième section, quelques développements plus récents sont présentés, notamment certains opérateurs de recombinaison et de mutation avancés. Finalement, des directions prometteuses pour des recherches futures sont proposées.

2. ALGORITHMES GÉNÉTIQUES

2.1. Généralités

Les algorithmes génétiques sont basés sur une analogie avec le phénomène de l'évolution et de la sélection naturelle. Le pionnier du domaine est John Holland [33] qui a le premier tenté d'implanter artificiellement des systèmes évolutifs basés sur le processus de la sélection naturelle. Par la suite, ces méthodes ont évoluées vers des algorithmes robustes d'optimisation. Afin de comprendre l'analogie entre le processus de sélection naturelle et l'algorithme d'optimisation que nous allons présenter, rappelons tout d'abord les mécanismes les plus importants qui sont à la base de l'évolution.

1. L'évolution se produit sur des chromosomes qui représentent chacun des individus dans une population.

2. Le processus de sélection naturelle fait en sorte que les chromosomes les mieux adaptés se reproduisent plus souvent et contribuent davantage aux populations futures.

3. Lors de la reproduction, l'information contenue dans les chromosomes des parents est combinée et mélangée pour produire les chromosomes des enfants (« croisement »).

4. Le résultat du *croisement* peut à son tour être modifié par des perturbations aléatoires (mutations).

1. Construire une population initiale de N chromosomes.

2. Évaluer chacun des chromosomes de la population.

3. Reproduire les chromosomes en sélectionnant les parents de façon proportionnelle à leur évaluation. Appliquer les opérateurs de croisement et de mutation lors de la reproduction.

4. Lorsque N nouveaux individus ont été générés, ils remplacent alors l'ancienne population. Les chromosomes de la nouvelle population sont évalués à leur tour.

5. Si le temps alloué n'est pas dépassé (ou le nombre de générations maximal n'est pas atteint), retourner à l'étape 3.

Figure 1. – Un algorithme génétique.

Ces mécanismes peuvent être utilisés pour spécifier un algorithme génétique tel que celui décrit à la figure 1. Pour adapter un algorithme génétique à un problème d'optimisation combinatoire, il suffit alors de spécialiser certaines composantes à la structure particulière de ce dernier. En général, les décisions portent sur les aspects suivants :

Codage des solutions : il faut établir une correspondance entre les solutions du problème et les chromosomes.

Opérateurs génétiques : il faut définir des opérateurs de croisement et de mutation pour les chromosomes.

Évaluation des chromosomes : la mesure d'adaptation des chromosomes à leur environnement est donnée par une fonction appelée « force » qui est calculée par un processus plus ou moins complexe à partir de la fonction objectif.

2.2. Aspects d'implémentation communs à de nombreuses applications

Dans les prochaines sections, nous étudierons comment différents chercheurs ont spécialisé certaines composantes pour divers problèmes d'optimisation combinatoire. Nous insisterons alors davantage sur les aspects particuliers à chacun des problèmes considérés, c'est-à-dire le codage, les opérateurs génétiques et la définition de la force. Il existe toutefois certains aspects d'implantation communs à la plupart des algorithmes génétiques, que nous abordons brièvement. Pour plus de détails, le lecteur est invité à consulter les ouvrages [15, 30].

Il existe plusieurs moyens pour mettre en œuvre certains des mécanismes importants d'un algorithme génétique. Par exemple, la sélection des parents peut s'effectuer de diverses façons. Dans une population $P = \{x_1, x_2, ..., x_N\}$ où chaque individu est évalué par la fonction f, l'opérateur classique consiste à associer à chaque individu x_i une probabilité proportionnelle $p_i = \dfrac{f(x_i)}{\sum_{j \in P} f(x_j)}$ d'être sélectionnée. En pratique, on utilise souvent d'autres méthodes de sélection pouvant se paramétriser plus facilement. Ainsi, dans la sélection par tournoi, un ensemble d'individus est sélectionné aléatoirement, dont les meilleurs éléments seulement sont retenus. Pour choisir deux parents, on peut donc sélectionner aléatoirement 5 individus et ne garder que les deux meilleurs. Dans d'autres méthodes, les individus sont ordonnés selon leur valeur pour la fonction f, et la sélection s'effectue aléatoirement selon le rang occupé par chaque individu. On utilise alors une distribution biaisée en faveur des individus en tête de liste.

Un autre mécanisme important est le mode de remplacement des individus de la population. Dans le modèle original, chaque population de N individus est totalement remplacée à chaque génération. Plus récemment, des stratégies « élitistes » font en sorte d'assurer à chaque génération la survie des meilleurs individus de la population. C'est le cas entre autres des modèles « stationnaires » où un nombre restreint d'individus est remplacé à chaque génération. Dans ce cas, on ajoute souvent des mécanismes supplémentaires empêchant l'ajout d'individus identiques (doublons) dans la population. Des implantations plus évoluées existent également pour des architectures parallèles [48].

3. REPRÉSENTATION DES SOLUTIONS ET OPÉRATEURS GÉNÉTIQUES

Pour adapter un algorithme génétique à un problème particulier, il faut d'abord définir une méthode de codage pour les solutions. Traditionnellement,

les premiers algorithmes génétiques utilisaient une représentation binaire où chaque solution est codée sous la forme d'une chaîne de bits. Si cette représentation est tout à fait appropriée pour certains problèmes d'optimisation combinatoire, il est maintenant courant d'utiliser d'autres représentations qui tirent parti de la structure naturelle des solutions. Dans cette section, nous décrivons diverses adaptations d'algorithmes génétiques, utilisant plusieurs représentations et opérateurs génétiques.

3.1. Représentation binaire

Les premiers algorithmes génétiques utilisaient tous une représentation binaire où chaque solution s est codée sous la forme d'une chaîne de bits de longueur n, i.e. $s[i] \in \{0, 1\}$, $\forall i = 1, ..., n$. Pour cette représentation, l'opérateur de croisement classique à un point [33] consiste à choisir de façon uniforme une position de coupure aléatoire $l \in \{1, 2, ..., n-1\}$. Si les parents p_1 et p_2 ont été sélectionnés pour la reproduction, les solutions enfants e_1 et e_2 sont alors construites de la façon suivante :

$$e_1[i] = p_1[i],\ \forall i \in \{1, ..., l\};\ e_1[i] = p_2[i],\ \forall i \in \{l+1, ..., n\};$$

$$e_2[i] = p_2[i],\ \forall i \in \{1, ..., l\};\ e_2[i] = p_1[i],\ \forall i \in \{l+1, ..., n\};$$

L'opérateur à un point peut se généraliser en sélectionnant k positions aléatoires plutôt qu'une. Dans ce cas, les enfants sont produits en échangeant l'information chromosomique entre les k points de coupure. En pratique, la valeur de $k = 2$ est la plus couramment utilisée. Un autre opérateur très populaire a été proposé par Syswerda [53]. Il s'agit de l'opérateur de croisement uniforme où une chaîne de bits est générée aléatoirement et est utilisée pour engendrer les enfants. Pour chaque position, le choix aléatoire indique quel parent déterminera la valeur des enfants. Des exemples des opérateurs de croisement à un point, à deux points, et uniforme sont illustrés dans les figures 2, 3 et 4. Dans [53], Syswerda présente des résultats théoriques et empiriques qui indiquent que l'opérateur de croisement uniforme donne généralement des résultats supérieurs à ceux produits par les deux autres. Avec une représentation binaire, on utilise généralement l'opérateur de mutation classique qui consiste à changer aléatoirement un bit pour son complément.

Parent 1	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**1**
Parent 2	0	1	0	1	1	1	0	1	1
Position aléatoire			↑						
Enfant 1	**1**	**0**	**0**	1	1	1	0	1	1
Enfant 2	0	1	0	**1**	**0**	**1**	**1**	**0**	**1**

Figure 2. – Opérateur de croisement à un point.

Parent 1	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**1**
Parent 2	0	1	0	1	1	1	0	1	1
Positions aléatoires			↑			↑			
Enfant 1	**1**	**0**	0	1	1	1	**1**	**0**	**1**
Enfant 2	0	1	**0**	**1**	**0**	**1**	0	1	1

Figure 3. – Opérateur de croisement à 2 points.

Parent 1	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**1**
Parent 2	0	1	0	1	1	1	0	1	1
Chaîne aléatoire de bits	0	1	1	0	1	0	1	0	0
Enfant 1	**1**	1	0	**1**	1	**1**	0	**0**	**1**
Enfant 2	0	**0**	**0**	1	**0**	1	**1**	1	1

Figure 4. – Opérateur de croisement uniforme.

3.1.1. Problème de satisfiabilité

Pour certains problèmes, la représentation binaire des solutions est tout à fait intuitive et bien adaptée. C'est le cas par exemple du problème de satisfiabilité (SAT). Étant donné une expression logique F de n variables booléennes $x_1, ..., x_n$, ce problème fondamental consiste à trouver une affectation aux variables de façon à satisfaire F. En établissant les correspondances *faux* $\Leftrightarrow$ 0, *vrai* $\Leftrightarrow$ 1, on peut représenter chaque solution par une chaîne binaire de longueur n et utiliser les opérateurs génétiques classiques décrits plus haut. Pour évaluer les solutions, plusieurs possibilités existent. Dans [16], les auteurs proposent d'évaluer à 1 les solutions qui satisfont F et d'attribuer une valeur dans l'intervalle [0, 1) à chaque solution ne satisfaisant pas F. Dans [18], les expressions booléennes sont transformées

sous forme conjonctive normale. Sous cette forme, l'expression logique F est composée d'une conjonction de m clauses

$$F = \bigwedge_{j=1}^{m} C_j.$$

Pour satisfaire F, il faut donc satisfaire toutes les clauses. Dans ce cas, chaque solution peut être simplement évaluée par le nombre de clauses qu'elle satisfait.

Dans [16], certains tests sont effectués sur des formules de taille modérée. Pour ces formules, les auteurs utilisent l'opérateur de croisement à deux points, et comptent le nombre moyen de solutions qui doivent être générées par un algorithme génétique avant de rencontrer une solution satisfaisant la formule considérée. On compare alors ces résultats au nombre moyen de solutions aléatoires qu'il faut générer pour arriver au même résultat. On rapporte alors des résultats encourageants en observant que les algorithmes génétiques atteignent des solutions admissibles beaucoup plus rapidement que les procédures de recherche aléatoire. Dans [18], des algorithmes génétiques sont appliquées à des formules beaucoup plus difficiles pour lesquelles on remarque également une amélioration tangible par rapport à la recherche aléatoire. Dans ce cas cependant, d'autres méthodes heuristiques sont comparées. On constate alors que les algorithmes génétiques traditionnels sont dominés par plusieurs autres méthodes heuristiques. On remarque également une domination de l'opérateur de croisement uniforme, tel que suggéré par les travaux de Syswerda [53]. À titre d'illustration, la figure 5 indique l'évolution de divers algorithmes génétiques utilisant différents opérateurs de croisement pour une formule SAT. On y retrouve en abscisse le nombre de solutions générés par les différentes approches, et en ordonnée la meilleure solution rencontrée.

3.1.2. Clique maximale dans un graphe

La représentation binaire des solutions peut également être facilement utilisée pour d'autres problèmes d'optimisation combinatoire. Considérons un graphe non-orienté $G = (V, E)$. Une clique de G est un sous-ensemble $C \subset V$ tel que $u, v \in C \Rightarrow (u, v) \in E$. Il est bien connu que le problème de trouver une clique de cardinalité maximale dans G est NP-complet [21]. Dans [6], un algorithme génétique est utilisé pour traiter ce problème. Dans cette adaptation, une solution potentielle est codée par une chaîne de bits de longueur $|V|$. Chaque chaîne s correspond alors à un sous-ensemble $V_s \subset V$

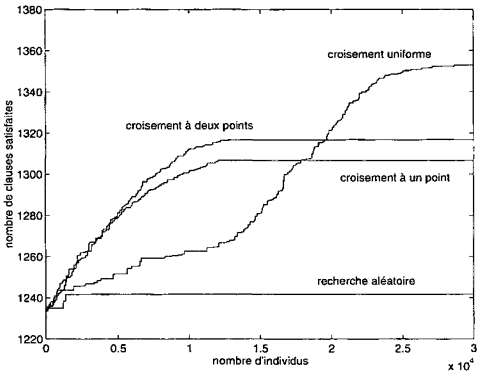

Figure 5. – Effet de différents opérateurs de croisement sur une formule SAT en forme conjonctive normale (349 variables, 1392 clauses). Population de taille 500, taux de mutation de 4 %. Moyenne de trois exécutions.

où $i \in V_s$ si et seulement si $s[i] = 1$. Les solutions sont alors évaluées de la façon suivante :

$$f(s) = \begin{cases} k \text{ si } V_s \text{ est une clique de taille } k \\ 0 \text{ sinon.} \end{cases}$$

Dans [6], un algorithme génétique utilisant cette représentation et l'opérateur de croisement à un point est expérimenté sur un échantillon de problèmes couramment utilisés pour comparer différentes approches heuristiques. Malgré une implantation parallèle sur une Connection Machine possédant 128 processeurs, l'algorithme génétique ne parvient pas à approcher les résultats rapportés avec d'autres méthodes [22, 18]. Nous verrons toutefois à la Section 3.3 que la méthode de codage et la fonction d'évaluation pour ce problème peuvent être améliorées et produire de meilleurs résultats.

3.1.3. Partionnement de graphe

Un autre problème classique d'optimisation combinatoire est celui du partitionnement de graphe. Étant donné un graphe $G = (V, E)$, on cherche une partition de V en deux ensembles V_1 et V_2, avec $|V_1| = |V_2| = |V|/2$. La partition optimale est alors celle qui minimise le nombre d'arêtes de E ayant leurs extrémités dans des sous-ensembles différents. Dans [5], Bui et Moon adaptent un algorithme génétique pour ce problème en utilisant une représentation binaire. Chaque chaîne s de longueur $|V|$ correspond à une partition (V_1, V_2), où $s[i] = 0 \Leftrightarrow i \in V_1$ et $s[i] = 1 \Leftrightarrow i \in V_2$. Les

solutions sont alors évaluées en comptant le nombre d'arêtes connectant les sous-ensembles V_1 et V_2.

Les auteurs utilisent un opérateur de croisement à 5 points de coupures choisis aléatoirement. Avec la fonction d'évaluation et l'opérateur de croisement utilisés, il faut cependant s'assurer que chaque chromosome contienne un nombre égal de bits à 1 et à 0. Après le croisement de deux solutions, les auteurs utilisent donc un mécanisme supplémentaire qui ajuste la solution résultante en changeant un nombre approprié de bits. L'opérateur de croisement compte d'abord la différence entre le nombre de 1 et de 0. Une position aléatoire est ensuite choisie, à partir de laquelle un certain nombre de valeurs sont modifiées jusqu'à l'obtention d'une solution réalisable. Un tel opérateur est souvent appelé croisement avec réparation. Dans ce cas particulier, un opérateur de recherche locale est également appliqué au chromosome produit par l'opérateur de croisement avec réparation. Les auteurs ont appliqué leur algorithme sur un ensemble important de problèmes tests et rapportent d'excellents résultats. En fait, leur implantation donne des résultats comparables ou meilleurs que ceux obtenus par une excellente adaptation du recuit simulé (*voir* [37]). Cependant, cette situation s'explique probablement par l'hybridation avec des méthodes de recherche locale. Cette approche sera d'ailleurs discutée plus en détail à la section 5.2.

3.1.4. Problèmes d'ordonnancement

Les représentations binaires ne sont pas limitées aux vecteurs à une dimension. Des structures plus complexes ont été récemment utilisées dans le cas de problèmes d'ordonnancement. Pour ces problèmes [3], il s'agit de déterminer un ordre de traitement pour n tâches. Il est alors possible de coder les solutions à l'aide d'une matrice de précédence composée de 0 et de 1. Nakano et Yamada [49] utilisent un croisement à un point sur la matrice de précédence limitée à sa partie (significative) triangulaire supérieure. Une procédure de réparation est ensuite appliquée de façon à obtenir une matrice complète transitive, antisymétrique et telle qu'il existe exactement une ligne et une seule contenant exactement k bits à 1 (pour k variant de 1 à n). Plus récemment [7], Portmann a défini un nouvel opérateur génétique de « type uniforme » utilisant directement les matrices de précédence de telle sorte que l'on puisse conserver le sous-ensemble de contraintes de précédence que l'on souhaite préserver. Ceci s'effectue soit au moyen d'un sous-ensemble imposé par le problème, soit par conservation : si i précède j chez les deux parents, alors i précède j chez les enfants créés. La matrice est ensuite complétée en utilisant une probabilité π de prendre un bit du parent 1 et

$(1 - \pi)$ du parent 2, chaque bit choisi provoquant plusieurs choix simultanés d'autres bits non encore fixés par transitivité.

3.2. Autres représentations

Pour les problèmes présentés dans la section précédente, l'utilisation d'une représentation binaire semble justifiée. Pour d'autres problèmes cependant, cette représentation est parfois contre-intuitive et semble peu appropriée. C'est pourquoi on a vu apparaître au cours des dernières années plusieurs algorithmes génétiques utilisant des systèmes de codage mieux adaptés aux structures naturelles des solutions.

3.2.1. Problème du commis-voyageur

Considérons le problème bien connu du commis-voyageur (TSP). Dans [53], la représentation binaire est utilisée pour coder une tournée passant par n villes. Chaque ville correspond à un groupe de bits qui représente un nombre en base 2. Ces valeurs sont triées et déterminent l'ordre de la tournée. Avec une telle représentation, on peut alors utiliser les opérateurs génétiques classiques décrits à la section précédente. Une autre école de chercheurs a proposé une représentation sous forme de permutation pour coder les solutions du TSP [29, 31]. Dans ce cas, chaque élément de la permutation donne directement l'ordre dans lequel les villes sont parcourues. Avec une telle représentation, il faut cependant construire de nouveaux opérateurs de croisement et de mutation. Il est en effet facile de constater que des opérateurs semblables à ceux décrits dans les figures 2, 3 et 4 ne pourraient préserver la structure de permutation. Plusieurs opérateurs ont toutefois été proposés, et sont décrits en détails dans [30, 31, 48, 46]. À titre d'exemple, l'opérateur OX de Davis [12, 51] choisit une sous-séquence dans un des parents, et place les villes restantes dans l'ordre défini par l'autre parent. Dans la figure 6, deux positions sont choisies aléatoirement afin de définir une sous-

Parent 1	5	1	2	7	4	3	6
Parent 2	2	1	4	3	6	7	5
Sous-séquence		↑			↑		
Étape 1		**1**	**2**	**7**	**4**		
Étape 2	6					5	3
Enfant	6	**1**	**2**	**7**	**4**	5	3

Figure 6. – Exemple de l'opérateur de croisement OX.

séquence dans le premier parent (étape 1). À l'étape 2, les villes restantes sont placées selon l'ordre induit par le deuxième parent, en commençant après le deuxième point de coupure. Un opérateur de mutation possible peut consister simplement à échanger deux éléments de la permutation.

Pour le TSP, des opérateurs de croisement plus sophistiqués existent également. L'opérateur de croisement par recombinaison d'arêtes [62, 63] tente de préserver le plus d'arêtes possible provenant des parents. Une « carte » d'arêtes contenant toutes les connexions entre les villes est construite pour les deux parents. Pour construire un enfant, une ville courante est choisie. La carte d'arêtes est alors utilisée pour choisir la prochaine ville courante. Parmi les villes liées à la ville courante dans un des parents, on choisit celle ayant elle-même le moins de liens disponibles dans la carte d'arêtes. Lorsqu'une nouvelle ville courante est ajoutée à la tournée, la carte d'arêtes est mise à jour. En général, ces adaptations donnent de bons résultats. Cependant, tel que discuté dans [31, 48], l'utilisation additionnelle d'une procédure de recherche locale est nécessaire pour générer des solutions de qualité comparable à celles obtenues par d'autres approches heuristiques [45].

3.2.2. Affectation quadratique

Des représentations par permutation ont également été adoptées pour d'autres problèmes d'optimisation combinatoire. Le problème d'affectation quadratique (QAP) peut s'exprimer comme

$$\min_{\phi \in P(n)} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}\, b_{\phi(i)\,\phi(j)}, \tag{1}$$

où $A = (a_{ij})$ et $B = (b_{kl})$ sont des matrices $n \times n$, et $P(n)$ est l'ensemble des permutations de $\{1, ..., n\}$. Une des applications importantes de ce problème consiste à affecter à moindre coût n objets à n sites. Dans ce cas, les matrices A et B sont symétriques, a_{ij} représentant le niveau d'interaction entre les objets i et j, et b_{kl} la distance entre les sites k et l. La contribution d'une affectation simultanée de l'objet i au site k et de l'objet j au site l est alors de $a_{ij}\, b_{kl}$. Le coût total est donc quadratique et peut être évalué par (1). Dans [59], des opérateurs de croisement et de mutation sont spécialement développés pour ce problème. Encore une fois cependant, les résultats rapportés dans [59] sont de beaucoup inférieurs à ceux obtenus avec d'autres méthodes itératives telles que celles décrites dans [56, 9].

3.2.3. Problèmes d'ordonnancement

Des représentations sous forme de permutation sont également utilisées pour plusieurs autres problèmes. En particulier, les problèmes d'ordonnancement constituent un domaine de recherche fertile pour les algorithmes génétiques. Pour ces problèmes, on cherche à déterminer un ordre de traitement pour un ensemble de tâches devant être traitées par un ensemble de processeurs. Il existe une multitude de variantes de ces problèmes (*voir* [3]), pour lesquels on peut considérer différents objectifs et contraintes régissant l'utilisation de ressources supplémentaires. Sauf exception [49, 7], on utilise dans la plupart des cas un codage sous forme de permutation avec des opérateurs génétiques adaptés. Parmi la littérature abondante sur ce sujet, notons [2, 7, 10, 13, 40 49, 54, 55, 60, 64].

3.2.4. Coloration de graphe

En plus des permutations, on retrouve maintenant une grande variété de méthodes de codage pour les solutions. Pour un graphe $G = (V, E)$, un coloriage est une partition de V en k sous-ensembles (couleurs) $C_1, C_2, ..., C_k$ pour lesquels $u, v \in C_i \Rightarrow (u, v) \notin E$. Le problème de coloration de graphe consiste à trouver un coloriage utilisant un nombre minimal de sous-ensembles. Dans [19], une valeur de k est prédéterminée et les solutions sont codées sous forme de chaînes de longueur $|V|$, avec $s[i] \in \{1, ..., k\}$, $\forall i \in V$. Chaque chromosome définit alors une partition de V en k sous-ensembles $V_1, ..., V_k$ par la relation $i \in V_j \Leftrightarrow s[i] = j$. Cette partition ne représente pas nécessairement un coloriage de G, et est évaluée par

$$f(V_1, V_2, ..., V_k) = \sum_{i=1}^{k} |E(V_i)| \tag{2}$$

où $E(V_i) = \{(x, y) \in E : x, y \in V_i\}$. On cherche alors à générer une partition avec $f(V_1, V_2, ..., V_k) = 0$.

Avec cette représentation, il est facile de définir des opérateurs de croisement et de mutation semblables à ceux utilisés pour les représentations binaires. Les résultats rapportés dans [19] indiquent que l'opérateur de croisement uniforme domine encore ceux à un point et à deux points (voir *fig.* 7).

3.3. Choix de la représentation

Plusieurs autres représentations sont maintenant utilisées pour divers problèmes d'optimisation combinatoire. Pour certains problèmes, il n'est

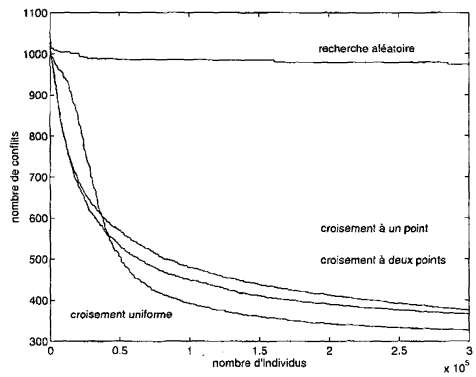

Figure 7. – Effet de différents opérateurs de croisement sur un graphe de Leighton (450 sommets, nombre chromatique : 15). Population de taille 400, taux de mutation de 1 %. Moyenne de trois exécutions.

pas toujours évident de choisir la représentation la plus appropriée. On peut d'ailleurs trouver dans [42] une discussion sur certains enjeux théoriques reliés à ce choix. Pour le problème du commis-voyageur par exemple, des représentations sous forme matricielle sont proposées dans [46], et des opérateurs génétiques sont spécialement définis pour cette représentation.

Pour le problème de la clique maximale décrit à la section 3.1.2, une représentation non-binaire est utilisée dans [18]. Dans ce cas, une clique de taille k est recherchée dans un graphe $G = (V, E)$. Chaque solution potentielle est alors représentée par un sous-ensemble $C \subset V$, où $|C| = k$. Un individu est alors évalué par

$$f(C) = \frac{(k-1)\,k}{2} - |E(C)| \tag{3}$$

$f(C)$ représente simplement le nombre d'arêtes manquantes pour que $(C, E(C))$ soit un graphe complet. Pour identifier une clique de taille k, il s'agit alors de trouver un individu de valeur nulle. Avec cette représentation, l'opérateur de croisement suivant est utilisé : étant donné deux parents C_1 et C_2 sélectionnés pour la reproduction, un enfant C_e est simplement généré en choisissant aléatoirement k sommets distincts dans $C_1 \cup C_2$. L'opérateur de mutation remplace simplement un nœud $x \in C_e$ par un autre $y \in V - C_e$. Cette nouvelle adaptation offre l'avantage de pouvoir être combinée avec d'autres méthodes. En effet, une procédure de recherche avec tabous décrite en [20] utilise la même représentation et évalue les solutions potentielles à l'aide de l'équation (3).

Dans la plupart des cas, le choix de la meilleure méthode de codage pour les solutions demeure un art, et peu de travaux se sont intéressés à définir des critères rigoureux pouvant guider ces choix [42]. Davis [15] suggère toutefois d'emprunter les systèmes de codage employés par d'autres méthodes couramment utilisées pour le problème considéré.

4. ÉVALUATION DE LA DIVERSITÉ D'UNE POPULATION

Lors des différentes phases d'un algorithme génétique, les populations sont plus ou moins variées. En général, la population initiale est composée d'individus très différents les uns des autres. Par la suite, durant la phase d'évolution, la reproduction plus fréquente des individus les plus performants produit des populations de plus en plus homogènes. On dit que l'algorithme génétique a convergé lorsque tous les individus d'une population sont presque identiques. Il est alors généralement inutile de poursuivre la phase d'évolution.

Il est souvent très utile de pouvoir mesurer de façon précise le degré de diversité d'une population. Cette mesure peut alors servir à définir des critères d'arrêt et aider à ajuster la valeur de divers paramètres (taille de la population, taux de mutation, sélection des parents, etc.) [4]. Dans [31], Grefenstette propose une mesure basée sur la notion d'entropie [61]. Cette notion est habituellement utilisée pour évaluer certaines distributions de probabilité. Avec cette mesure, l'entropie d'une distribution est maximale lorsque tous les événements possibles sont équiprobables. Parmi les distributions uniformes, l'entropie est plus élévée lorsque plusieurs états sont possibles.

Une mesure de diversité basée sur la notion d'entropie peut être adaptée pour plusieurs représentations. Pour le problème d'affectation quadratique par exemple, les solutions sont codées sous forme de permutation, et l'entropie d'une population P peut alors être évaluée par

$$E = \frac{-\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\frac{n_{ij}}{|P|}\right)\log\left(\frac{n_{ij}}{|P|}\right)}{n\log n}, \tag{4}$$

avec la convention que $0 \log 0 = 0$ et où n_{ij} représente le nombre de fois où l'objet i est affecté au site j dans la population P. Avec cette valeur normalisée, $E \in [0, 1]$. Une valeur $E = 0$ indique que la population est constituée d'individus identiques, alors qu'une valeur près de 1 suggère une grande diversité dans la population.

Une mesure similaire peut aussi être utilisée dans le cas d'une représentation binaire (pour le problème SAT, par exemple). Définissons n_{ij} comme le nombre de fois où la position i est fixée à j (0 ou 1) dans la population P. Une mesure de diversité basée sur la notion d'entropie est alors donnée par

$$E = \frac{-\sum_{i=1}^{n} \sum_{j=0}^{1} \frac{n_{ij}}{|P|} \log \frac{n_{ij}}{|P|}}{n \log 2}, \tag{5}$$

avec la convention que $0 \log 0 = 0$. Comme dans le cas précédent, $E \in [0, 1]$ indique jusqu'à quel point les bits sont distribués uniformément dans la population.

La disponibilité d'une mesure de diversité de la population peut être utile dans plusieurs cas. Tout d'abord, elle permet d'étudier le comportement d'un algorithme génétique utilisant différents opérateurs ou valeurs des paramètres. Par exemple, la figure 8 illustre le comportement de la diversité d'une population pour les différents opérateurs de croisement utilisés pour le problème de coloration de graphe de la figure 7.

Dans la figure 8, on note que l'opérateur de croisement uniforme est celui qui préserve le mieux la diversité de la population, et qu'il produit les meilleurs résultats. Des mécanismes supplémentaires sont d'ailleurs souvent utilisés pour préserver la diversité des populations. Par exemple, il est courant [15, 17] dans un algorithme génétique stationnaire d'empêcher l'ajout

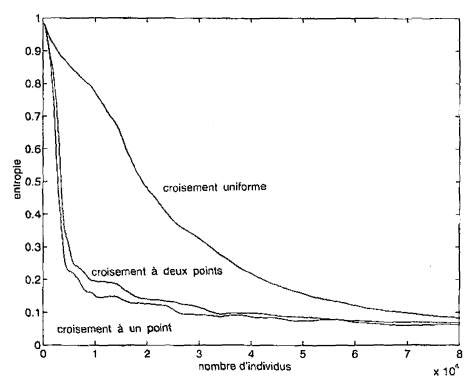

Figure 8. – Effet de différents opérateurs de croisement sur la diversité d'une population. Coloration d'un graphe de Leighton (450 sommets, nombre chromatique : 15). Population de taille 400, taux de mutation de 1 %. Moyenne de trois exécutions.

d'individus identiques (doublons) dans la population. On peut également ajuster la valeur de certains paramètres selon la diversité de la population. Entre autres, les taux de mutation et de croisement, ainsi que les paramètres régissant la sélection des parents peuvent être ajustés selon la mesure de diversité de la population.

5. OPÉRATEURS GÉNÉTIQUES AVANCÉS

Comme nous l'avons vu dans la dernière section, des algorithmes génétiques ont déjà été adaptés pour plusieurs problèmes d'optimisation combinatoire. Dans la plupart des cas cependant, les implantations traditionnelles ne peuvent rivaliser avec d'autres approches heuristiques. Des développements récents semblent toutefois indiquer que certaines approches hybrides peuvent s'avérer très compétitives, et parmi les plus performantes pour certains problèmes d'optimisation combinatoire. Ces approches hybrides utilisent le plus souvent des opérateurs de croisement spécialisés pour le problème traité, ainsi que des opérateurs de mutation avancés tels que ceux décrits dans cette section.

5.1. Croisement adapté

Les opérateurs de croisement classiques décrits à la section 3.1 ont le grand avantage d'être très robustes. On peut en effet les utiliser pour tout problème dont on peut coder les solutions sous forme binaire. Ce souci de robustesse a longtemps constitué un objectif majeur pour les algorithmes génétiques. On s'est cependant rapidement rendu compte qu'il était avantageux de spécialiser ces opérateurs pour les problèmes particuliers. Un premier pas dans cette direction fut l'utilisation de représentations non-binaires telles que celles décrites dans la section 3.2. Dans [31], Grefenstette a été un peu plus loin en utilisant un opérateur de croisement spécialement adapté au problème de commis-voyageur. Dans son implantation, un codage sous forme de permutation est utilisé, et l'opérateur de croisement considère l'information relative aux distances entre les villes lorsque plusieurs villes sont candidates à occuper une position. Cette approche contraste avec l'algorithme classique où les opérateurs sont habituellement « aveugles » et n'utilisent l'information reliée au problème que lors de l'évaluation des chromosomes. Grefenstette rapporte une amélioration tangible des résultats obtenus.

Il est possible de construire des opérateurs de croisement semblables pour plusieurs autres problèmes d'optimisation combinatoire. Pour le problème de

satisfiabilité par exemple, l'opérateur décrit à la figure 9 est utilisé dans [18]. On considère ici une formule sous la forme conjonctive normale, *i.e.* une conjonction de clauses où chaque clause est une disjonction de littéraux. Pour chaque clause C_j, on définit C_j^+ comme l'ensemble des littéraux positifs de la clause j; *i.e.* $x_i = 1$, $i \in C_j^+ \Rightarrow C_j$ est évaluée à vrai. C_j^- est l'ensemble des littéraux négatifs de la clause C_j; *i.e.* $x_i = 0$, $i \in C_j^- \Rightarrow C_j$ est évaluée à vrai. Une formule SAT en forme conjonctive normale peut alors s'exprimer sous la forme

$$F = \bigwedge_{j=1}^{m} C_j, \text{ où } C_j = \Big(\bigvee_{i \in C_j^+} x_i \Big) \vee \Big(\bigvee_{i \in C_j^-} (1 - x_i) \Big). \tag{6}$$

Pour i=1 à n **faire**
  enfant[i]=–1
Pour $j \in \overline{S}$(*parent*$_1$) **faire**
  Si ($j \in S$(*parent*$_2$))
    Pour $i \in C_j^+ \cup C_j^-$ **faire**
      enfant[i] = *parent*$_2$[i]
Pour $j \in \overline{S}$(*parent*$_2$) **faire**
  Si ($j \in S$(*parent*$_1$))
    Pour $i \in C_j^+ \cup C_j^-$ **faire**
      enfant[i]=*parent*$_1$[i]
Pour i=1 à n **faire**
  Si (*enfant*[i]=–1)
    Si (*DU*[0, 1]=1)
      enfant[i]=*parent*$_1$[i]
    Sinon
      enfant[i]=*parent*$_2$[i]

Figure 9. – Croisement adapté à une formule SAT.

Pour décrire l'opérateur de croisement adapté, on introduit également la notation suivante. Pour une solution s, $S(s) \in \{1, ..., m\}$ est l'ensemble des clauses satisfaites, et $\overline{S}(s)$ l'ensemble des clauses insatisfaites. DU [0, 1] est une fonction qui retourne 0 ou 1 avec la même probabilité. La procédure décrite à la figure 9 décrit alors un opérateur de croisement spécialement

adapté pour le problème de satisfiabilité. Cet opérateur recherche les clauses qui sont satisfaites dans un parent, mais qui ne le sont pas dans l'autre. On affecte aux variables impliquées dans une telle clause les mêmes valeurs que celles du parent dont la clause est satisfaite. Les variables non affectées par ce processus sont traitées comme dans le cas de l'opérateur de croisement uniforme. Tel qu'illustré à la figure 10, l'utilisation d'un tel opérateur est habituellement de beaucoup préférable aux opérateurs classiques décrits à la section 3.1.

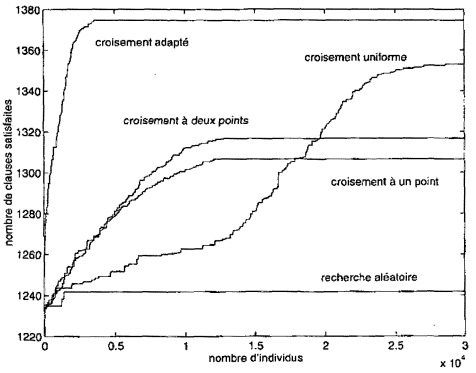

Figure 10. – Effet d'un opérateur de croisement spécialement adapté pour une formule SAT en forme conjonctive normale (349 variables, 1392 clauses). Population de taille 500, taux de mutation de 4 %. Moyenne de trois exécutions.

Pour la coloration de graphe, certains opérateurs de croisement adaptés à la structure du graphe sont proposés dans [18, 19]. Pour ces opérateurs, le principe est d'identifier des nœuds conflictuels (de même couleur et reliés par une arête) dans un parent, et de les colorier comme dans l'autre parent s'ils y sont coloriés sans conflit. Comme dans le cas de la satisfiabilité, ces opérateurs peuvent permettre la convergence vers de bonnes solutions en nécessitant moins de générations. Ils requièrent cependant plus de temps que les opérateurs classiques. C'est pourquoi ils sont particulièrement intéressants lorsque combinés avec des opérateurs de mutation avancés tels que ceux décrits dans la section 5.2. Dans ce cas, la plupart de l'effort de calcul est consacré à la mutation, et le temps requis par l'opérateur de croisement devient négligeable.

Dans le cas d'opérateurs de croisement adaptés, il est probablement plus approprié de les désigner de façon générale comme des opérateurs de recombinaison. Dans ce cas en effet, on peut alors englober des opérateurs

similaires issus d'autres écoles que celle des algorithmes génétiques [23, 28]. Pour certains problèmes pratiques comportant plusieurs contraintes supplémentaires, la conception d'un opérateur de recombinaison peut devenir très complexe. C'est le cas par exemple pour le problème de confection d'un calendrier de ligue sportive présenté dans [11]. Dans cette application, les solutions potentielles doivent satisfaire un ensemble de contraintes diverses. L'opérateur de recombinaison proposé doit donc posséder des mécanismes supplémentaires afin de rétablir certaines propriétés pour les solutions enfants. Dans [11], Costa parvient à obtenir, non sans difficultés, un opérateur satisfaisant.

5.2. Mutation avancée

Bien que l'utilisation d'opérateurs de croisement spécialisés améliore de beaucoup la qualité des résultats obtenus, ils ne sont souvent pas suffisants pour rendre les algorithmes génétiques compétitifs avec d'autres approches heuristiques [18]. Cette situation change toutefois lorsque les mutations sont effectuées par des opérateurs plus puissants, tels que des méthodes de descente ou de recherche avec tabous.

5.2.1. Opérateurs de recherche locale

Une procédure de recherche locale consiste à examiner le voisinage d'une solution courante et à progresser d'une solution à l'autre lorsque se produit une amélioration de l'objectif. Historiquement, ces méthodes de descente ont toujours compté parmi les méthodes heuristiques les plus populaires pour traiter les problèmes d'optimisation combinatoire [43, 44]. Il n'est donc pas étonnant de constater que plusieurs chercheurs les utilisent maintenant à titre d'opérateurs de mutation. Muhlenbein [48] et Davis [15] sont parmi les premiers à avoir combiné algorithmes génétiques et procédures de recherche locale, même si des idées semblables ont été émises beaucoup plus tôt dans un contexte plus général [23].

De telles approches hybrides produisent généralement d'excellents résultats. En fait, les approches génétiques et de recherche locale semblent complémentaires. Le principal défaut des méthodes de descente réside dans le fait qu'elles doivent s'interrompre lorsqu'aucune solution voisine ne peut améliorer la solution courante. Pour pallier à cette lacune, une mesure couramment utilisée consiste à répéter le processus à partir de plusieurs solutions de départ générées aléatoirement. Or il s'avère que cette façon de procéder engendre de bien meilleurs résultats lorsque les points de départs sont fournis par un algorithme génétique.

On trouve des exemples de tels algorithmes pour plusieurs problèmes d'optimisation combinatoire, dont ceux du commis-voyageur [48, 31], du partitionnement de graphe [5], de l'affectation quadratique [17], de la coloration de graphe [19], de la clique maximale, et de la satisfiabilité [18]. À titre d'exemple, la figure 11 illustre le comportement d'un algorithme génétique hybride pour un problème d'affectation quadratique tiré de [9]. Dans ce cas, une procédure de recherche locale est appliquée à chaque enfant produit par un algorithme génétique sous-jacent. On remarque une amélioration évidente de l'algorithme génétique hybride lorsque comparé à l'approche de redémarrage aléatoire.

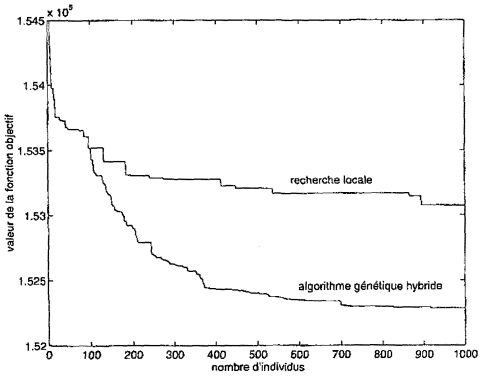

Figure 11. – Effet bénéfique d'opérateurs génétiques sur la recherche locale pour un problème d'affectation quadratique ($n = 100$). **Population de taille 100, Moyenne de cinq exécutions.**

5.2.2. Opérateurs de recherche avec tabous

Les méthodes de recuit simulé [38, 1] et de recherche avec tabous [25, 26, 27] sont des améliorations de la méthode de recherche locale. Il est donc naturel d'imaginer de les utiliser comme opérateurs de mutation dans le cadre d'approches génétiques hybrides. Un hybride combinant un algorithme génétique et une méthode de recuit simulé a d'ailleurs été proposé dans [34] pour le problème d'affectation quadratique. Cependant, on y rapporte peu de résultats numériques et il est difficile de juger de la performance relative de l'approche. À notre avis, la méthode de recherche avec tabous nous semble mieux adaptée pour remplir le rôle d'opérateur de mutation dans le cadre d'approches génétiques hybrides. En effet, contrairement au recuit simulé, elle ne nécessite pas la définition d'un programme de refroidissement qui peut induire des calculs très longs pour obtenir des solutions de qualité.

On compte déjà quelques adaptations très puissantes combinant opérateurs génétiques et méthode de recherche avec tabous. Dans [17], on rapporte d'excellents résultats pour le problème d'affectation quadratique. En fait, cette approche a permis d'améliorer les meilleures solutions connues de plusieurs exemplaires tests classiques pour ce problème. Des expérimentations ont également été complétées pour les problèmes de la clique maximale et de la satisfiabilité [18], de même que pour celui de la coloration de graphe [19]. À titre d'exemple, la figure 12 illustre le comportement d'un algorithme hybride pour la coloration d'un graphe de Leighton [41]. Pour cet hybride, une procédure de recherche avec tabous est appliquée à chaque individu. L'individu résultant de cette recherche est ensuite ajouté à la population pour la génération suivante. Dans ce cas, l'approche hybride permet de colorier le graphe en 15 couleurs, alors que la méthode de recherche avec tabous seule n'y parvient qu'en utilisant 16 couleurs. Les opérateurs génétiques ont alors un effet diversificateur bénéfique et améliorent ainsi une méthode heuristique déjà très performante. Cet objectif de diversification constitue actuellement un domaine de recherche très important pour les méthodes de recherche avec tabous. Les succès obtenus dans [17, 18, 19] semblent indiquer un avenir intéressant pour cette voie de recherche. On peut d'ailleurs apparenter de telles méthodes hybrides à l'idée de « path relinking » émise dans [28].

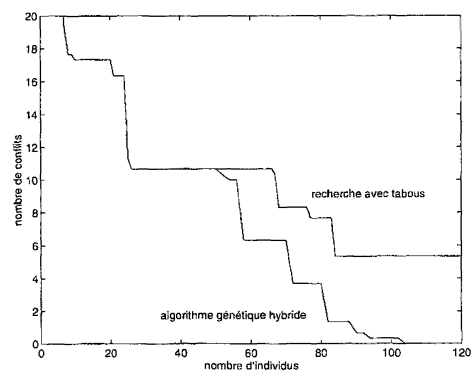

Figure 12. – Algorithme génétique hybride utilisant une mutation de 10 000 itérations de recherche avec tabous sur un graphe de Leighton (450 sommets, nombre chromatique : 15). Population de taille 50, moyenne de trois exécutions.

Ces résultats sont très encourageants mais doivent toutefois être mis en perspective. Lorsqu'on applique un opérateur de recherche avec tabous à chaque individu généré par un algorithme génétique hybride, les temps de

traitement augmentent très rapidement. Pour le problème de la coloration de graphe, des algorithmes génétiques hybrides [19] ont réussi à colorier certains graphes très difficiles qui n'ont pu être coloriés par d'autres méthodes heuristiques. Pour d'autres graphes cependant, une méthode de recherche avec tabous bien adaptée parvient à obtenir des coloriages de qualité équivalente en beaucoup moins de temps que pour les algorithmes génétiques hybrides.

Lorsque des opérateurs de mutation avancés sont utilisés, les résultats obtenus sont habituellement de très grande qualité, mais les temps nécessaires pour atteindre la convergence peuvent devenir prohibitifs. On peut trouver une discussion dans ce sens dans [58]. Dans cet article, plusieurs approches sont comparées pour un large éventail de problèmes d'affectation quadratique. On y conclut que les approches génétiques hybrides sont parmi les plus puissantes, mais également les plus coûteuses en temps. Le choix de la méthode à utiliser dépend donc en grande partie du temps de traitement disponible pour résoudre un problème considéré. Pour améliorer les temps de traitement, les recherches futures devraient se concentrer sur des implantations parallèles ou une utilisation plus parcimonieuse des opérateurs de mutation avancés [47].

6. CONCLUSION

En général, les algorithmes génétiques sont considérés comme des méthodes « faibles » dans le sens qu'elles nécessitent peu d'information sur le problème à optimiser. En fait, une méthode de codage, des opérateurs de croisement et de mutation, ainsi qu'une fonction d'évaluation sont suffisants pour utiliser une approche évolutive. On obtient alors une approche robuste, mais qui est souvent dominée par des méthodes spécialisées aux problèmes particuliers. Dans le cas de problèmes d'optimisation combinatoire cependant, il existe souvent des structures particulières permettant le développement d'approches hybrides menant à des algorithmes heuristiques très puissants, souvent parmi les meilleurs disponibles [48, 5, 17, 19, 18].

Malgré ces résultats encourageants, beaucoup de travail reste à faire pour améliorer les performances de telles approches évolutives. Le principal objectif consiste à obtenir une convergence plus rapide vers des solutions de qualité. Heureusement, plusieurs voies de recherches sont ouvertes dans cette direction. À notre avis, la recherche future devrait viser des implantations parallèles efficaces [48], des opérateurs de recombinaison améliorés, et une utilisation stratégique d'opérateurs d'optimisation supplémentaires [47, 28].

RÉFÉRENCES

1. E. Aarts et J. Korst, *Simulated Annealing and Boltzmann Machines,* Wiley, 1989.
2. S. Bagchi, S. Uckun, Y. Miyabe et K. Kawamura, Exploring Problem-Specific Recombination Operators for Job Shop Scheduling, *Proceedings of the Fourth International Conference on Genetic Algorithms,* San Mateo, Calif., Morgan Kaufmann Publishers, 1991, p. 10-17.
3. J. Blazewicz, W. Cellary, R. Slowinski et J. Weglarz, Scheduling Under Resource Constraints: Deterministic Models, *Annals of Operations Research, 7,* 1986.
4. L. Booker, Improving Search in Genetic Algorithms, dans [14], 1987, p. 61-73.
5. T. N. Bui et B. R. Moon, *A Genetic Algorithm for a Special Class of the Quadratic Assignment Problem, Special Issue on Quadratic Assignment and Related Problems,* DIMACS Series, 1993 (à paraître).
6. B. Carter et K. Park, How good are genetic algorithms at finding large cliques: an experimental study, *Technical Report BU-CS-93-015,* Boston University, 1994.
7. C. Caux, H. Pierreval et M. C. Portmann, Les algorithmes génétiques et leur application aux problèmes d'ordonnancement, *Actes des Journées d'Étude Ordonnancement et entreprise : applications concrètes et outils pour le futur,* CNRS, Toulouse, 1994, p. 5-45.
8. V. Cerny, A Thermodynamical Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm, *Journal of Optimization Theory and Applications,* 1985, *15,* p. 41-51.
9. J. Chakrapani et J. Skorin-Kapov, Massively parallel tabu search for the quadratic assignment problem, *Annals of Operations Reseach,* 1993, *41,* p. 327-341.
10. G. A. Cleveland et S. F. Smith, Using Genetic Algorithms to Schedule Flow-Shop Releases, *Proceedings of the Third International Conference on Genetic Algorithms,* Morgan Kaufmann Editors, Los Altos, CA, 1989, p. 160-169.
11. D. Costa, An Evolutionary Tabu Search Algorithm and the NHL Scheduling Problem, *INFOR,* 1995, *33,* p. 161-178.
12. L. Davis, Applying Adaptive Algorithms to Epistatic Domains, *Proceedings of the International Joint Conference on Artificial Intelligence,* 1985, p. 162-164.
13. L. Davis, Job-Shop Scheduling With Genetic Algorithms, *Proceedings of the First International Conference on Genetic Algorithms,* J. J. Grefenstette, éditeur, Lawrence Erlbaum Associates, Hillsdale, NJ, 1985, p. 136-140.
14. L. Davis, *Genetic Algorithms and Simulated Annealing,* Morgan Kaufmann Publishers, 1987.
15. L. Davis, *Handbook of Genetic Algorithms,* Van Nostrand Reinhold, New York, 1991.
16. K. A. DeJong et W. M. Spears, Using Genetic Algorithms to Solve NP-Complete Problems, *Proceedings of the Third International Conference on Genetic Algorithms,* Morgan Kaufmann Editors, Los Altos, CA, 1989, p. 124-132.
17. C. Fleurent et J. A. Ferland, Genetic Hybrids for the Quadratic Assignment Problems, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science,* (P. M. Pardalos et H. Wolkowicz eds.), 1994, *16,* p. 173-188.
18. C. Fleurent et J. A. Ferland, Object-Oriented Implementation of Heuristic Search Methods for Graph Coloring, Maximum Clique, and Satisfiability, *DIMACS series in Discrete Mathematics and Theoretical Computer Science,* (M. Trick et D. Johnson eds.), 1996 (à paraître).
19. C. Fleurent et J. A. Ferland, Genetic Algorithms and Hybrids for Graph Coloring, *Annals of Operations Research,* 1995, *60.*
20. C. Friden, A. Hertz et D. De Werra, STABULUS: A Technique for Finding Stable Sets in Large Graphs with Tabu Search, *Computing,* 1989, *42,* p. 35-44.

21. M. R. Garey et D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* W. H. Freeman and Co., San Francisco, 1979.

22. M. Gendreau, P. Soriano et L. Salvail, Solving the Maximum Clique Problem Using a Tabu Search Approach, *Annals of Operations Research*, 1993, *41*, p. 385-403.

23. F. Glover, Heuristics for Integer Programming Using Surrogate Constraints, *Decision Sciences,* 1977, *8*, p. 156-166.

24. F. Glover, Future Paths for Integer Programming and Links to Artificial Intelligence, *Computer and Operations Research,* 1986, *13*, p. 533-549.

25. F. Glover, Tabu Search-Part I, *ORSA Journal on Computing,* 1989, *1*, p. 190-206.

26. F. Glover, Tabu Search-Part II, *ORSA Journal on Computing,* 1990, *2*, p. 4-32.

27. F. Glover et M. Laguna, Tabu search, dans *Modern Heuristic Techniques for Combinatorial Problems,* C. R. Reeves éditeur, Blackwell Scientific Publications, Oxford, 1993, p. 70-141.

28. F. Glover, *Genetic Algorithms and Scatter Search: Unsuspected Potentials,* Rapport technique, University of Colorado at Boulder, 1993.

29. D. E. Goldberg et R. Lingle, Alleles, Loci, and the TSP, *Proceedings of the First International Conference on Genetic Algorithms,* J. J. Grefenstette éditeur, Lawrence Erlbaum Associates, Hillsdale, NJ, 1985, p. 154-159.

30. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

31. J. J. Grefenstette, Incorporating Problem Specific Knowledge into Genetic Algorithms, dans [14], 1987, p. 42-60.

32. A. Hertz et D. De Werra, Using Tabu Search Techniques for Graph Coloring, *Computing,* 1987, *39*, p. 345-351.

33. J. H. Holland, *Adaptation in Natural and Artificial Systems,* Ann Arbor, University of Michigan Press, 1975.

34. C. L. Huntley et D. E. Brown, A Parallel Heuristic for Quadratic Assignment Problems, *Computers and Operations Research*, 1991, *18*, p. 275-289.

35. T. Ibaraki, Enumerative Approaches to Combinatorial Optimization, Part I, *Annals of Operations Research,* 1987, *10*.

36. T. Ibaraki, Enumerative Approaches to Combinatorial Optimization, Part II, *Annals of Operations Research*, 1987, *11*.

37. D. S. Johnson, C. R. Aragon, L. A. McGeoch et C. Schevon, Optimization by Simulated Annealing: An Experimental Evaluation, Part I, Graph Partioning, *Operations Research*, 1989, *37*, p. 865-892.

38. S. Kirkpatrick, C. D. Gelatt et M. P. Vecchi, Optimization by Simulated Annealing, *Science,* 1983, *220*, p. 671-680.

39. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan et D. B. Shmoys, *The Traveling Salesman Problem: A guided Tour of Combinatorial Optimization*, John Wiley and Sons, New York, 1985.

40. S. Lash, *Genetic Algorithms for Weighted Tardiness Scheduling on Parallel Machines,* Technical Report 93-01, Department of Industrial Engineering and Management Sciences, North Western University, 1993.

41. F. Leighton, A graph coloring algorithm for large scheduling problems, *Journal of Research of the National Bureau of Standards*, 1979, *84*, p. 489-505.

42. G. E. Liepins et M. D. Vose, Representational Issues in Genetic Optimization, *Journal of Experimental and Theoretical Artificial Intelligence,* 1990, *2*, p. 101-105.

43. S. Lin, Computer Solutions of the Traveling Salesman Problem, *Bell System Technical Journal,* 1965, *44*, p. 2245-2269.

44. S. LIN et B. W. KERNIGHAN, An Effective Heuristic for the Traveling Salesman Problem, *Operations Research*, 1973, *21*, p. 498-516.

45. M. MALEK, M. GURUSWAMY, M. PANDYA et H. OWENS, Serial and Parallel Simulated Annealing and Tabu Search Algorithms for the Traveling Salesman Problem, *Annals of Operations Research*, 1989, *21*, p. 59-84.

46. Z. MICHALEWICZ, *Genetic Algorithms+Data Structures=Evolution Programs*, Springer-Verlag, Berlin, 1992.

47. P. MOSCATO, An introduction to population approaches for optimization and hierarchical objective functions: A discussion on the role of tabu search, *Annals of Operations Research*, 1993, *41*, p. 85-121.

48. H. MUHLENBEIN, M. GORGES-SCHLEUTER et O. KRAMER, Evolution Algorithms in Combinatorial Optimization, *Parallel Computing*, 1988, *7*, p. 65-88.

49. R. NAKANO et T. YAMADA, Conventional Genetic Algorithm for Job Shop Problems, *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, Calif., Morgan Kaufmann Publishers, 1991, p. 474-479.

50. R. NELSON et R. J. WILSON, *Graph Colourings*, Pitman Research Notes in Mathematics Series, *218*, Longman Scientific and Technical, Harlow, Essex, UK, 1990.

51. I. M. OLIVER, D. J. SMITH et J. R. C. HOLLAND, A Study of Permutation Crossover Operators on the Traveling Salesman Problem, *Proceedings of the Second International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, 1987, p. 224-230.

52. P. M. PARDALOS et H. WOLKOWICZ, *Quadratic Assignment and Related Problems*, DIMACS Series in Mathematics and Theoretical Computer Science, *16*, édité par l'American Mathematical Society, 1994.

53. G. SYSWERDA, Uniform Crossover in Genetic Algorithms, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, Calif., Morgan Kaufmann Publishers, 1989, p. 2-9.

54. G. SYSWERDA, Schedule Optimization Using Genetic Algorithms, dans [15], 1991, p. 332-349.

55. G. SYSWERDA et J. PALMUCCI, The Application of Genetic Algorithms to Resource Scheduling, *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, Calif., Morgan Kaufmann Publishers, 1991, p. 502-507.

56. E. TAILLARD, Robust Taboo Search for the Quadratic Assignment problem, *Parallel Computing*, 1991, *17*, p. 443-455.

57. E. TAILLARD, *Recherches itératives dirigées parallèles*, thèse de doctorat, École Polytechnique Fédérale de Lausanne, 1993.

58. E. TAILLARD, *Comparison of iteratives searches for the quadratic assignment problem*, Rapport technique ORWP94/04, DMA, École Polytechnique Fédérale de Lausanne, 1994.

59. D. E. TATE et A. E. SMITH, A genetic approach to the quadratic assignment problem, *Computers and Operations research* (à paraître).

60. S. UCKUN, S. BAGCHI et K. KAWAMURA, Managing Genetic Search in Job Shop Scheduling, *IEEE Expert*, 1993, *8*, p. 15-24.

61. D. WELSH, *Codes and Cryptography*, Oxford University Press, New York, 1988.

62. D. WHITLEY, T. STARKWEATHER et D. FUQUAY, Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator, *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Editors, Los Altos, CA, 1989, p. 133-140.

63. D. WHITLEY, T. STARKWEATHER et D. SHANER, The Traveling Salesman and Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination, dans [15], 1991, p. 350-372.

64. T. YAMADA et R. NAKANO, A Genetic Algorithm Applicable to Large Scale Job Shop Problems, dans *Parallel Problem Solving from Nature, 2*, R. MANNER et B. MANDERICK éds., North-Holland, Amsterdam, 1992, p. 281-290.