Génération et contrôle autonomes d'opérateurs pour les algorithmes évolutionnaires

Jorge Maturana, Frédéric Lardeux, Frédéric Saubion

Génération et contrôle autonomes d'opérateurs pour les algorithmes évolutionnaires

Jorge Maturana Frédéric Lardeux Frédéric Saubion

LERIA, Université d'Angers
{nom}@info.univ-angers.fr

Résumé

Les algorithmes évolutionnaires ont largement démontré leur utilité pour la résolution de problèmes combinatoires. Toutefois, leur utilisation pratique suppose de régler, d'une part, un certain nombre de paramètres fonctionnels et, d'autre part, de définir judicieusement les opérateurs qui seront utilisés pour la résolution. En effet, comme pour la majorité des méthodes métaheuristiques, les performances d'un algorithme évolutionnaire sont intrinsèquement liées à sa capacité à correctement gérer l'équilibre entre l'exploitation et l'exploration de l'espace de recherche. Récemment, de nouvelles approches ont vu le jour pour rendre ces algorithmes plus autonomes, notamment en automatisant le réglage et/ou le contrôle de paramètres. Nous proposons ici une nouvelle méthode dont l'objectif est double : d'une part nous souhaitons contrôler dynamiquement le comportement des opérateurs au sein d'un algorithme génétique, à travers leurs probabilités d'application et, d'autre part, nous souhaitons gérer un ensemble important d'opérateurs potentiels, dont l'utilisateur ne connaît pas a priori les performances, de manière également automatisée. Grâce à un mécanisme d'évaluation de l'état de la recherche en cours et de récompenses et de pénalités, le système devra identifier les opérateurs efficaces au détriment de ceux qui le sont moins. Nous expérimentons cette approche sur le problème SAT afin de démontrer qu'un algorithme autonome peut obtenir des performances similaires à celles d'un algorithme dédié, disposant d'opérateurs spécifiquement sélectionnés. Cette démarche vise finalement à libérer l'utilisateur de tâches fastidieuses de réglage et de l'expertise nécessaire à la conception d'algorithmes, souvent ad-hoc.

Abstract

Evolutionary algorithms have been efficiently used for solving combinatorial problems. However, their practical use induces to fix a set of functional parameters and also to define carefully the suitable operators for the resolution. As with the majority of metaheuristics methods, the performance of an evolutionary algorithm is intrinsically linked to its ability to properly manage the balance between exploitation and exploration of search space. Recently, new approaches have emerged to make these algorithms more independent, especially by automating the setting and / or control of parameters. We propose a new approach whose objective is twofold : on one hand we want to dynamically control the behavior of operators in a genetic algorithm, thanks to their probabilities of application and, on the other hand, we want to manage an important set of potential operators, whose performances are a priori unknown. Using a mechanism for evaluating the current state of search and a system of rewards and penalties, we identify the efficient operators and the bad ones. We test this methodology on the SAT problem to demonstrate that an algorithm can autonomously performs similar to a dedicated algorithm, whose operators have been specifically designed. This approach actuially aims to free the user from tedious setup tasks and from the often ad-hoc expertise, which is needed for the design of such solving algorithms.

1 Introduction

Les algorithmes évolutionnaires (AE) ont largement été utilisés pour l'optimisation discrète et continue, balayant un large spectre d'applications. Basés initialement sur les principes de l'évolution naturelle, les AEs permettent de gérer un ensemble de configurations d'un problème, modifiées progressivement au moyen d'opérateurs de variation et ce, afin de converger progressivement vers une solution optimale ou sous-optimale de bonne qualité. La métaphore évolutionniste amène à considérer ces configurations comme des individus formant une population qui évolue au moyen d'opérateurs génétiques, de mutation ou de croisement, selon leurs spécificités. Ce cadre général de résolution de problèmes s'inscrit dans le contexte des méthodes dites métaheuristiques et les principales

difficultés liés à la mise en oeuvre pratique de tels algorithmes reposent sur le choix d'un codage adéquat du problème ainsi que sur la définition d'opérateurs efficaces. Un fois cette architecture définie, le comportement de l'algorithme est en général déterminé par un ensemble de paramètres qui permettent d'agir principalement sur ses capacités à correctement explorer et exploiter l'espace de recherche, c'est à dire l'ensemble des configurations possibles du problème. Nous choisissons ici de distinguer les paramètres structurels de l'AE, tels que la taille de la population, des paramètres comportementaux, tels que les probabilités d'application des différents opérateurs. Le réglage de ces paramètres [11] s'avère alors être une tâche particulièrement ardue qui repose sur le savoir-faire, bien souvent empirique, de l'utilisateur et sur ses connaissances des caractéristiques du problème à résoudre. Dès lors, l'ajustement de paramètres s'appuie sur une série, en général coûteuse, d'expériences. Une telle approche réduit considérablement l'universalité des valeurs obtenues et la transposition de ces expérimentations à d'autres problèmes. Une autre voie consiste alors à envisager un contrôle des paramètres durant l'exécution de l'AE. Ce contrôle peut être supervisé par un ensemble de connaissances acquises au préalable (par exemple, choix d'une méthode de résolution dans les algorithmes de type *portfolio*) ou encore être abordé dans une optique plus autonome, les paramètres évoluant en fonction de l'état courant de la recherche. Ce nouveau paradigme visant à produire des algorithmes de résolution autonomes est en pleine émergence, à l'intersection de plusieurs domaines de l'informatique, notamment en intégrant des outils issus de l'apprentissage automatique, de l'optimisation combinatoire ou encore de la programmation par contraintes [1, 2, 9]. Mais, outre le contrôle des paramètres, le choix même des composants structurels de l'AE nécessite également une expertise de la part de l'utilisateur car, si des opérateurs de variations standard existent dans la littérature (comme le croisement uniforme par exemple), l'obtention de résultats acceptables est inévitablement conditionné par la spécialisation du schéma algorithmique général et la définition d'opérateurs appropriés.

Dans ce contexte, notre objectif est de proposer une nouvelle approche pour rendre les AE plus autonomes, à la fois dans le choix des opérateurs qu'ils utilisent mais également dans le réglage des paramètres qui leur sont intrinsèquement liés. Se basant sur des travaux précédents permettant de contrôler dynamiquement les probabilités d'application des opérateurs d'un AE [14, 12], notre approche peut être résumée par la figure 1.

Les performances de l'AE doivent être évaluées en fonction de mesures permettant de rendre compte de

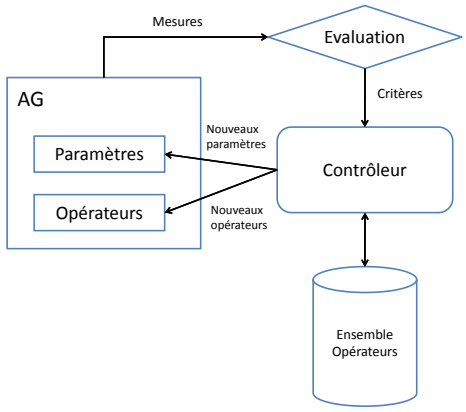

FIG. 1 – Schéma général

l'état de la recherche en cours. Deux notions sont traditionnellement mises en avant comme les clés du succès de ces techniques de recherche heuristiques de solutions dans un espace potentiellement très vaste : la diversification et l'intensification. Alors que la diversification rend compte de l'aptitude de la méthode à explorer des zones variées de l'espace de recherche, l'intensification correspond à la propension qu'a l'algorithme à converger vers une solution dans une zone précise. Dans le cadre des AE, nous mettons en avant deux mesures que nous avons précédemment utilisées pour contrôler cet équilibre entre intensification et diversification [13] : la qualité moyenne de la population et sa diversité (ici mésurée comme l'entropie des valeurs de vérité affectées aux variables). Ces deux mesures seront alors utilisées par un processus d'évaluation afin que le contrôleur puisse, d'une part, déterminer les nouvelles valeurs des paramètres et, d'autre part, choisir les opérateurs à conserver dans l'AE.

Dans cette nouvelle conception de l'AE, nous disposons d'un ensemble initial (qui pourrait également être généré dynamiquement par une "fabrique d'opérateurs") d'opérateurs qui peuvent être inclus dans l'AE à tout moment, leur taux d'application étant fixé par un paramètre idoine. Par conséquent, le principe général du contrôle consiste à distinguer les "bons" opérateurs, pour les appliquer davantage et favoriser leur action bénéfique, des "mauvais" pour les remplacer au sein de l'AE par de nouveaux candidats. La complexité de cette approche réside de manière duale dans l'évaluation de la situation en cours et dans la manière de récompenser ou de pénaliser les opérateurs.

Dans cet article, nous appuierons notre validation expérimentale sur la résolution du problème du problème canonique de satisfaction en logique propositionnelle (SAT) [7, 17]. Par le passé, nous avons pro-

posé un AE dédié au problème SAT (GASAT [10]) qui s'est avéré performant lors de la campagne d'évaluation SAT 2004. Forts de cette expérience, nous définissons ici un ensemble suffisamment large d'opérateurs de croisements (plus de 300) susceptibles d'être utilisés au sein d'un même AE, intégrant un mécanisme de contrôle autonome des probabilités d'applications de ces croisements. Notre objectif est double :

- Montrer que l'AE peut sélectionner de manière autonome des opérateurs adaptés à l'état courant de la recherche grâce au mécanisme de contrôle que nous lui adjoignons, et,
- Montrer que, sans connaissance préalable des opérateurs les plus performants, nous pouvons réobtenir, grâce au mécanisme de contrôle, des résultats comparables à ceux que nous avions obtenus avec notre algorithme GASAT, qui bénéficiait d'une analyse approfondie des meilleurs opérateurs possibles.

2 Contrôle autonome de paramètres pour les AEs

De nombreux travaux ont abordé le problème du réglage de paramètres pour les algorithmes évolutionnaires et nous renvoyons le lecteur à [11] pour un panorama plus exhaustif. Dans ce chapitre, nous nous concentrerons sur les éléments directement liés à la méthode que nous développons ici.

2.1 Réglage de paramètres pour les AEs

Le réglage de paramètre peut être abordé en se référant à la taxonomie proposée par Eiben et al. [4].

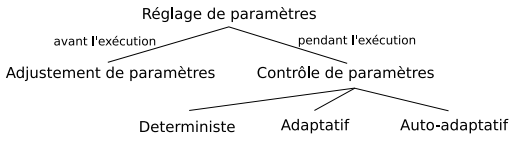

Fig. 2 – Taxonomie du contrôle de paramètre proposée par Eiben et al. [4]

Dans cette classification, on distingue les stratégies de réglage suivant qu'elles cherchent à fixer les valeurs des paramètres avant l'exécution de l'algorithme ou pendant celle-ci. Dans le cas d'un paramétrage initial, on cherche avant tout à trouver des valeurs universelles qui s'appliqueront sur la classe de problèmes la plus large possible. De tels résultats supposent de collecter un nombre important de mesures expérimentales et reposent en général sur une observation empirique ou, tout au moins, une extrapolation de phénomènes observés. S'agissant d'un réglage effectué pendant l'exécution même de l'algorithme, les auteurs distinguent ici trois branches selon le degré d'autonomie de la stratégie employée. Les approches déterministes reposent sur l'utilisation de règles prédéfinies alors que les approches adaptatives et auto-adaptatives cherchent à modifier les paramètres en fonction de l'observation de l'état courant du processus de recherche.

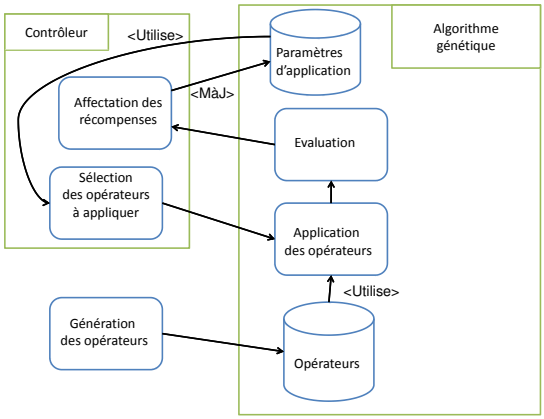

Fig. 3 – Approche générale pour la sélection adaptative d'opérateurs

Nous nous intéressons ici à la branche adaptative, qui, de manière plus ambitieuse, vise à libérer au maximum l'utilisateur du réglage et de la compréhension, plus ou moins intuitive, de l'impact des paramètres sur la recherche.

2.2 Sélection adaptative d'opérateurs (SAO)

Nous voulons décrire à présent les principes de bases qui ont émergé ces dernières années pour concevoir des algorithmes évolutionnaires capables de gérer leurs opérateurs de manières plus autonomes. Le schéma de la figure 3 permet de distinguer les deux phases essentielles, utilisées dans les approches qui nous intéressent ici : la récompense des opérateurs et la sélection de ces mêmes opérateurs relativement à la manière dont ils ont été récompensés..

La récompense, matérialisée par un crédit qui sera affecté aux opérateurs, s'appuie sur une évaluation de l'état courant de la recherche. La plupart des méthodes s'appuient souvent sur une utilisation exclusive de la qualité des populations (moyenne, meilleur individu...) [18, 5]. Dans des travaux précédents, nous avons proposé d'utiliser conjointement la qualité moyenne de la population et sa diversité [13]. Une fois les critères d'évaluation fixés, l'affectation de crédits aux opérateurs peut alors être plus où moins sophistiquée (utilisation de fenêtres de valeurs, calcul de valeurs moyennes ou utilisation de valeurs extrêmes). L'objec-

tif principal est de proposer le système de récompenses le plus générique et le plus robuste possible mais qui soit également suffisamment sensible aux changements d'états afin de permettre au mécanisme de sélection d'opérateur d'être réactif. Les crédits peuvent être vus comme un moyen d'apprendre dynamiquement des informations sur la qualités des opérateurs au cours de la résolution.

La sélection d'opérateurs va s'appuyer sur les récompenses obtenues précédemment pour indiquer à l'algorithme quel(s) opérateur(s) utiliser lors du prochain état basique de calcul, réglant ainsi les paramètres initiaux de l'AE. Ici encore, les choix sont multiples allant d'une probabilité d'application proportionnelle à la récompense à des modèles plus sophistiqués, issus de la théorie des jeux. La question principale est de proposer un système de contrôle qui permette de rester réactif et adaptatif, ne convergeant pas trop vers une utilisation systématique d'opérateurs au comportement "moyen".

3 Opérateurs de croisements pour SAT

3.1 Le problème SAT

Le problème SAT [7, 17] consiste à trouver une affectation satisfaisant une expression Booléenne. Une instance de ce problème est donc une formule Booléenne qui peut être mise sous forme normale conjonctive (CNF), c'est à dire sous la forme d'une conjonction de clauses où les clauses sont des disjonctions de littéraux (variables ou négations de variable). Quand toutes les clauses peuvent être satisfaites, le problème est dit satisfiable. Dans le cas contraire, il est souvent intéressant de minimiser le nombre de clauses fausses. Deux familles de méthodes permettent de résoudre ces problèmes : les méthodes exactes qui répondent au problème de décision (satisfiable ou non satisfiable) et les méthodes approchées qui répondent au problème d'optimisation (minimiser le nombre de clauses fausses).

3.2 Algorithmes évolutionnaires pour SAT

Les AEs pour SAT [3, 6, 8, 16, 10] font partie des méthodes approchées. Comme cela a été rappelé dans l'introduction, le principe de base de ces algorithmes est de manipuler une population d'individus à l'aide d'opérateurs génétiques afin d'obtenir des individus de bonne qualité. Dans le cas du problème SAT, les individus sont des affectations des variables Booléennes (i.e., d'interprétations logiques possibles) et l'objectif est d'avoir des individus induisant le moins de clauses fausses possibles. Plusieurs types d'opérateurs sont utilisés au sein des AEs : sélection, croisement, mutation, insertion ...

L'algorithme génétique pour SAT que nous allons utiliser pour nos expériences repose sur GASAT [10] qui est actuellement l'un des AEs les plus efficaces pour SAT. Son principe basique est le suivant :

1. une population est aléatoirement générée ;
2. deux individus sont sélectionnés aléatoirement dans une sous population des 15 meilleurs individus ;
3. l'opérateur de croisement CC (défini section 3.3) est appliqué sur ces individus ;
4. l'opérateur de mutation applique une recherche locale sur le fils obtenu ;
5. le nouvel individu est inséré dans la population et remplace le plus vieil individu s'il est meilleur que le moins bon individu de la sous population ;
6. si aucune des conditions d'arrêt (affectation satisfaisant le problème, nombre de croisements maximum atteint, ...) n'est atteinte, retour à 2.

L'objectif de notre travail étant de mettre en avant les propriétés du contrôleur, nous avons modifié GASAT pour n'en garder que le squelette et ainsi observer la seule action du contrôleur. Pour cela, nous avons remplacé l'opérateur de sélection par une sélection aléatoire et nous avons supprimé la mutation. L'opérateur d'insertion est quant à lui modifié pour qu'il substitue automatiquement, au plus vieil individu, le fils nouvellement créé.

3.3 Une famille d'opérateurs de croisement

L'opérateur de croisement est un opérateur très important. Il permet à la population d'être régénérée avec de bons individus. Dans notre problématique de contrôle autonome, la définition d'un bon individu dépend de l'état de la recherche. Dans certains cas, il peut être plus intéressant pour la recherche d'obtenir un individu qui diversifie la population et dans d'autres, les individus améliorant la qualité seront les plus attendus. Pour cette raison, nous avons décidé de travailler avec plusieurs croisements ayant chacun une prédisposition, soit à améliorer la qualité, soit à favoriser la diversité. La majorité des croisements existants essaie de conserver les bonnes propriétés des parents afin de les reproduire chez le fils. Par exemple :

- le croisement uniforme conserve les valeurs de vérité des variables qui sont identiques chez les deux parents ;
- le croisement proposé par Fleurent et Ferland [6] utilise l'ensemble des clauses qui sont vraies chez un parent et fausses chez l'autre. Seules sont conservées les valeurs des variables apparaissant dans les clauses vraies de cet ensemble ;

– le croisement CC [10] ne traite que les clauses fausses simultanément chez les deux parents et les rend vraies chez le fils en flippant une variable dans chacune d'elles ;
– le croisement CCTM [10] opère de la même manière que CC mais il travaille ensuite sur les clauses vraies chez les deux parents afin de garantir que les clauses seront aussi vraies chez le fils.

Très peu d'opérateurs de croisements ont comme objectif principal la diversification. Nous en avons donc redéfini plusieurs qui essaient de détecter les bonnes propriétés des parents pour ensuite les casser. Par exemple :
– le croisement uniforme "inverse" qui flippe toutes les valeurs de vérité des variables qui sont identiques chez les deux parents ;
– le croisement NT ne traite que les clauses vraies chez les deux parents et les rend fausses chez le fils.

Nous avons proposé et utilisons beaucoup d'autres croisements (307 au total) qui sont des combinaisons de croisements existants. Dans cet ensemble se trouve des croisements dont l'objectif principal est de diversifier, d'autres dont la tâche première est d'améliorer la qualité et enfin un grand nombre dont l'action est moins tranchée.

4 Description du contrôleur

Dans cette section nous allons décrire précisément le mécanisme de contrôle de l'AE que nous proposons. L'architecture globale de notre approche est détaillée par la figure 4, sur laquelle on constate que le contrôleur comprend principalement deux composants :
– la sélection adaptative d'opérateur (SAO) qui, comme présentée dans la section 2, constitue l'interface de commande permettant de déterminer quel est l'opérateur à appliquer (*Sélection d'opérateurs* (SO)) et de collecter la rétroaction de l'AE, c'est à dire l'évaluation de l'impact de l'opérateur sur la recherche en cours afin de lui attribuer une récompense (*Affectation de crédits* (AC)).
– Le *forgeron* qui est responsable de fournir des opérateurs adéquats pour l'AE. Ce forgeron se base sur une spécification générale de schémas d'opérateur et sur les diverses options possibles que ces opérateurs peuvent combiner.

Bien que l'on puisse considérer que ces composants contrôlent tous deux des paramètres de l'AE, il existe pourtant une différence fondamentale entre leurs actions respectives. Tandis que la SAO contrôle des paramètres comportementaux (i.e., choisir la probabilité d'application d'un opérateur à un moment donné), le forgeron modifie des paramètres structurels (choisir quels opérateurs seront insérés). Les choix de la SAO sont donc dépendants de ceux du forgeron.

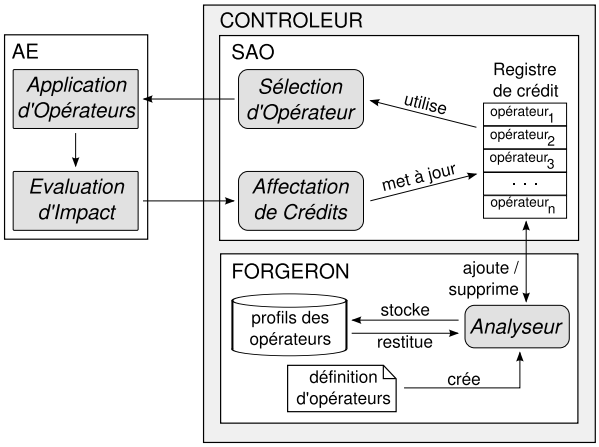

FIG. 4 – Schéma général du contrôleur

4.1 Affectation de crédits

Comme mentionné dans la section 2, et afin d'évaluer la performance des opérateurs, il faut mesurer leurs performances après application. La méthode présentée dans [12, 14] prend en compte trois mesures différentes : la variation de diversité, la variation de qualité et le temps d'exécution. L'objectif est alors de maximiser les deux premiers critères et de minimiser le troisième. Comme ces objectifs sont contradictoires, nous avons besoin de trouver un façon de les combiner pour obtenir une évaluation unique.

La méthode *Compass (C)*, que nous avions proposé dans [14] (Figure 5.a), prend en considération la distance d'un point $(\Delta Diversité, \Delta Qualité)$, représentant l'évaluation d'un opérateur o, à une ligne incliné d'un angle $\Theta = \pi/4$. Le point le plus à droite et en haut est le meilleur. Puis, ces valeur comparatives sont divisées par le temps d'exécution pour récompenser les opérateurs les plus rapides.

Dans cet article, deux autres principes de mesure ont été comparés, tous deux basés sur la notion de dominance Pareto [15]. De manière succincte, lorsque deux points sont comparés suivant plusieurs critères ($\Delta Diversité$ et $\Delta Qualité$ dans notre cas), on dit qu'un point *domine* l'autre s'il est meilleur (plus grand dans notre cas) sur tous les aspects. Dans le cas contraire, les points sont incomparables.

L'affectation de crédits dénommée *Dominance Pareto (DP)* considère le nombre d'opérateurs que chaque opérateur domine (Figure 5.b) ; par contre, pour la mesure *Rang Pareto (RP)*, on regarde combien d'opérateurs dominent chaque opérateur (les va-

leurs les plus petites sont alors préférables). Une différence importante existe entre ces deux évaluations. Alors que le RP considère uniquement des opérateurs qui ne sont pas dominés, la DP récompense de ceux qui sont en forte concurrence avec d'autres opérateurs. Un effet de type essaim se produit ici : il ne suffit pas d'être bon mais aussi il faut être bon là où la plupart des opérateurs sont placés.

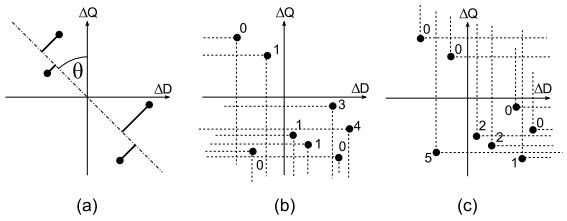

FIG. 5 – Schémas d'affectation de crédits. Compass (a), Dominance Pareto (b), Rang Pareto (c)

4.2 Sélection d'opérateurs

L'idée de la SO présentée dans [12], appelée Ex-DMAB, est inspirée des méthodes de bandits manchots (MAB) utilisées en théorie de jeux. Ici, une stratégie choisit toujours l'opérateur le plus performant, en même temps qu'elle assure de ne pas retarder indéfiniment l'invocation des autres opérateurs. Afin de s'adapter aux environnements dynamiques, comme c'est le cas pour les AEs, un test statistique surveille les changements des séries temporelles associées aux opérateurs, et réinitialise la mémoire du MAB lors d'un changement de comportement évident comme, par exemple, lorsque l'actuel meilleur opérateur est dépassé par un autre.

Dans le travail que nous présentons ici, nous comparons quatre SO différentes :

MAB2 , inspiré de Ex-MAB [5], mais adaptée pour bien fonctionner avec des ensembles dynamiques d'opérateurs. En effet, la méthode présentée dans [5] se base sur le nombre de fois où les opérateurs ont été utilisés pour n'"oublier" aucun d'entre eux. Ceci ne fonctionne pas bien lorsque de nouveaux opérateurs viennent d'être introduits dans l'AE, car il faudrait les appliquer un certain nombre de fois pour qu'ils rattrapent les autres. Nous proposons alors un mécanisme qui consiste à considérer le nombre de générations écoulées depuis la dernière application de chacun. De plus, pour des raisons de simplicité, le mécanisme de réinitialisation n'est pas utilisé.

Aléatoire (A) , choisit au hasard l'opérateur à appliquer.

MAB2+Détection de Collusion (M2C) , similaire à MAB2, il comporte en plus un mécanisme de détection de collusion des individus (nous choisissons ce terme, sans doute un peu fort, traduisant le regroupement autour des optima locaux, qui nuit à la poursuite de la recherche). Ceci est réalisé en considérant la pente de la ligne résultant de la régression linéaire des valeurs de qualité moyenne de la population. Si la pente est proche de zéro, on considère que l'AE est "coincé"' et on démarre une phase d'exploration forcée, en ne prenant que les opérateurs qui se sont avérés être exploratoires. Cette phase est abandonnée lorsque la diversité atteint une certaine fraction au-dessus de la diversité actuelle, quand il n'y a pas d'opérateurs de diversification, ou quand un nombre de générations sans pouvoir augmenter la diversité s'est écoulé.

Probabilité Proportionnelle (PP) choisit les opérateurs avec une probabilité proportionnelle aux récompenses enregistrées dans le registre de crédit.

4.3 Forgeron

Le Forgeron, comme évoqué en section 2, constitue la "fabrique d'opérateurs", qui contrôle alors les paramètres structurels correspondant aux opérateurs présents à chaque instant de l'exécution de l'AE. Il remplit les fonctions suivantes :

Crée (forge) des opérateurs, à partir de leur définition/spécification,

Ajoute les opérateurs au registre de crédit, en fournissant à la SAO de nouveaux choix d'opérateurs,

Analyse la performance des opérateurs dans le registre, pour éventuellement supprimer un opérateur de l'AE ou en ajouter un nouveau,

Supprime les opérateurs les moins performants en fonction de l'information stockée dans le registre de crédit,

Stocke le profil des opérateurs éliminés, afin d'avoir une trace de leur existence, pour éventuellement les ré-insérer dans le registre plus tard,

Restitue des opérateurs éliminés en cas de besoin.

Les opérateurs peuvent donc avoir trois états principaux : *non-nés*, avant qu'ils ne soient créés pour la première fois ; *vivants*, quand ils sont placés dans le registre de crédit et sont utilisés par la SAO, et *morts*, après avoir été éliminés du registre.

L'élimination d'un opérateur ne signifie nécessairement pas qu'il soit mauvais *per se*, mais plutôt dans le contexte des nécessités actuelles de la recherche : un opérateur peut être un excellent diversificateur, mais être éliminé si l'état actuel de la recherche a besoin d'opérateurs d'exploitation pour converger davantage.

Ceci explique pourquoi les profils des opérateurs sont stockés lorsqu'ils sont éliminés du registre : en cas de besoin, la résurrection d'un opérateur ne se fera pas aveuglément.

Dans notre implémentation, le registre de crédit est composé d'un nombre fixe d'opérateurs, et évalué à intervalles réguliers par le forgeron dans la recherche des opérateurs faibles qui puissent être éliminés pour laisser place à un nouveau né. Tous les opérateurs possibles sont créés puis essayés avant de rechercher à nouveau parmi les morts, ce qui permet de donner à tous une chance de démontrer leurs capacités.

5 Étude expérimentale

Nous présentons ici une analyse expérimentale de la méthode décrite plus haut, dans le cadre de la résolution du problème SAT.

5.1 Cadre d'application

Pour nos expériences nous avons sélectionné des instances venant de différentes compétitions SAT. Le choix des benchmarks essaie de couvrir les différentes familles d'instances (aléatoires, faites-main et industrielles) : *f500* est une instance générée aléatoirement au seuil ; *aim-100-1_6-yes1-1* (notée aim-100 par la suite) est une instance aléatoire modifiée afin de n'avoir qu'une seule solution ; *3bitadd_31.shuffled* (notée 3bitadd par la suite) est une instance de VLSI ; *ibm-2004-29-k55* (notée ibm par la suite) est une instance industrielle (BMC) ; *simon-s02b-r4b1k1.2* (notée simon par la suite) est une instance difficile des compétitions SAT.

L'algorithme de base servant à nos expériences (voir section 3.2) est appliqué 50 fois pour chaque croisement et chaque contrôleur et cela pour chacune des 5 instances testées. Lors d'une exécution, la population possède 100 individus et le nombre de croisements autorisé est de 10^5.

5.2 Réglage du contrôleur

L'objectif ici est d'essayer les différentes combinaisons de mécanismes d'affectation de crédits et de sélection d'opérateurs présentés en 4.1 et 4.2. Ces combinaisons seront identifiés par la notation $X - Y$, où $X \in \{C, DP, RP\}$ est le mécanisme d'AC, et $Y \in \{M2, A, M2C, PP\}$ est le mécanisme de SO (voir section 4).

Les paramètres du contrôleur sont ceux des SOs MAB2 et M2C ainsi que ceux du Forgeron. Du coté de MAB2, l'opérateur à appliquer sera celui qui maximise l'expression $MAB2_{o,t}$, correspondant à l'opérateur o à l'instant actuel t qui est donné par la formule suivante.

$$MAB2_{o,t} = r_{o,t} + 2 \cdot exp(p \cdot a_{o,t} - p \cdot x \cdot trc_t) \quad (1)$$

où $r_{o,t}$ est la récompense de l'opérateur o à l'instant t, correspondant à la valeur maximale des 10 dernières applications, $a_{o,t}$ est le temps d'*oubli* (i.e., le nombre de générations écoulées depuis la dernière application de o), trc_t correspond à la taille du registre de crédit à l'instant t. x est un paramètre qui indique combien de générations doit laisser passer un paramètre dans la phase d'oubli avant d'être forcement appliqué, exprimé comme un facteur de la taille du registre de crédit ($x > 1$). La force de cette exploration est exprimée par une exponentielle, qui n'a presque aucune importance au début, mais augmente rapidement à la fin de la période définie par x. p influence la croissance de cette exponentielle.

La formule 1 comporte deux parties. Celle de gauche encourage l'application des opérateurs qui ont reçu les meilleures récompenses, et celle du droite encourage les opérateurs qui n'ont pas été appliqués depuis longtemps. Les valeurs de $r_{o,t}$ sont normalisées dans l'intervalle $[0, 1]$. La partie de droite donne une valeur proche de zéro, sauf quand $a_{o,t}$ se rapproche de $x \times trc_t$, implicant, dans ce cas, une valeur de 2. En pratique, aucun opérateur n'est abandonné plus de $(x \times trc_t)$ générations. Les valeurs pour les paramètres lors de nos expériences étaient $trc_t = 20$, $x = 1.5$ et $p = 0.2$.

Dans MAB2+DC, on trouve de plus les paramètres de détection de collusion. La phase d'exploration est décrétée si dans les 100 dernières générations, la différence entre la plus grande et la plus petite des qualités moyennes de la population est inférieure à 0.001 et la pente de la ligne résultant de la régression linéaire est comprise dans ± 0.0001.

On pourrait objecter ici que nous remplaçons les paramètres initiaux d'application des opérateurs par de nouveaux paramètres tout aussi délicats à régler. Toutefois, nous avons clairement mesuré qu'un moins bon réglage du contrôleur avait un impact bien moindre sur la performance de l'algorithme qu'un mauvais réglage de ses paramètres comportementaux. D'autre part, nous travaillons ici avec quelques centaines d'opérateurs et donc quelque centaines de paramètres initiaux (sans même évoquer l'espace des algorithmes possibles induit par la combinaison de ces opérateurs), alors que nous devons fixer trois ou quatre paramètres pour le contrôleur.

En ce qui concerne le Forgeron, le registre de crédit a un taille fixe de 20 opérateurs. Toutes les 50 générations, le forgeron analyse le registre dans le but de trouver un opérateur faible pour le supprimer et le remplacer par un opérateur non-né ou mort. Si un opérateur a un nombre suffisant d'applications ($\frac{1}{2}$ de

la taille de la fenêtre, c'est à dire 5 applications) et que sa récompense est dans le tiers inférieur par rapport au reste, il est choisi pour être éliminé.

6 Résultats

Nous présentons ici le compte-rendu expérimental des résultats obtenus avec notre approche, en tentant d'en illustrer les aspects les plus frappants. La figure 6 montre la convergence moyenne sur 50 exécutions du meilleur individu pour les différents combinaisons d'AC et de SO pour l'instance *ibm*. L'abcisse correspond au générations écoulées (chaque génération correspondant à une application d'opérateur).

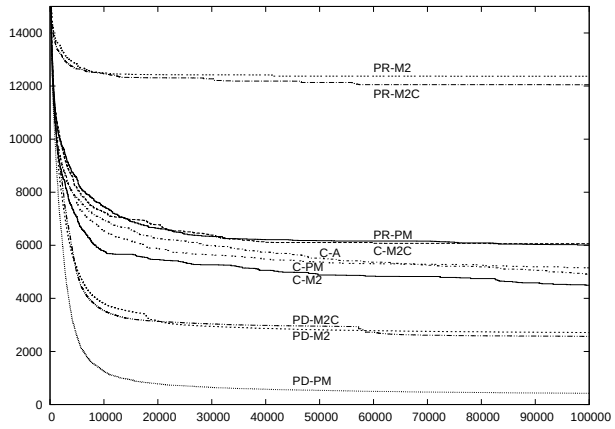

FIG. 6 – Nombre de clauses fausses du meilleur individu sur instance *ibm*, avec différents contrôleurs

Ici, les meilleurs résultats correspondent aux contrôleurs DP-PP, DP-M2 et DP-M2C. La figure 7 montre la diversité moyenne sur 50 exécutions pour ces contrôleurs sur la même instance. Notons que des contrôleurs qui obtiennent des niveaux similaires en terme de qualité, ne maintiennent pas forcement les mêmes niveaux de diversité. DP-PP se caractérise par son instabilité en terme de diversité, dûe à l'utilisation de la SO PP, qui, par rapport à M2 et M2C, induit une capacité d'exploration supérieure. Ceci explique la bonne performance de DP-PP sur toutes les instances, comme on le verra par la suite.

La tendance à la hausse de la diversité pour DP-PP à partir de la génération 8000 est dûe à la prise en compte de la diversité dans l'évaluation des opérateurs. Au début de la recherche, la population initiale est composée d'individus générés aléatoirement. Ceci facilite la tâche des opérateurs "exploitatoires", qui font décroître la diversité en même temps que la qualité augmente. Cependant, à partir d'un certain moment, l'amélioration devient difficile, et le contrôleur se tourne vers l'exploration, en cherchant à accroître la

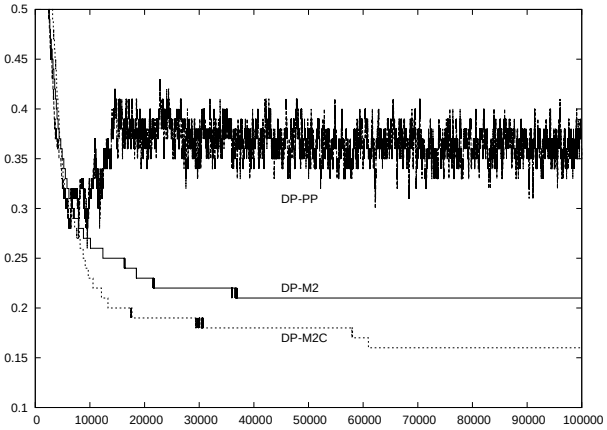

FIG. 7 – Diversité des trois meilleurs contrôleurs sur instance *ibm*

diversité afin que les individus puissent s'échapper des optima locaux. Ce comportement est précisément celui que l'on cherche en incluant la diversité dans l'évaluation qui est transmise au contrôleur.

Cette tendance est particulièrement forte dans l'AC C. La figure 8 montre l'évolution de la diversité moyenne sur 50 exécutions pour les différents méthodes d'AC : C, DP et RP utilisées conjointement avec la SO M2C, sur l'instance *simon*. Dans cette figure on peut apprécier la, sans doute, excessive exploration induite par C, ce qui expliquerait que ces résultats ne soient pas les meilleurs de la série.

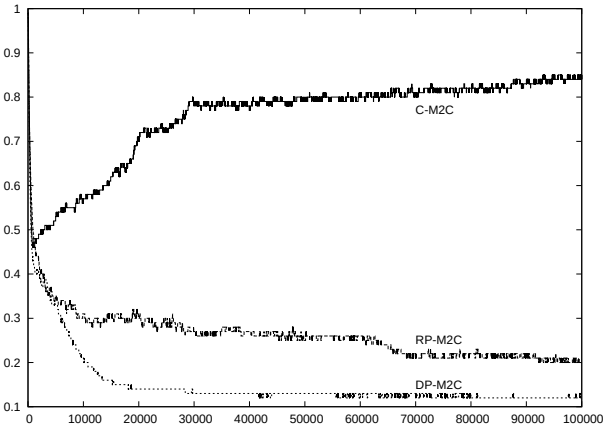

FIG. 8 – Diversité des CAs C, DP et RP avec SO M2C, montrant le passage de l'exploitation vers l'exploration avec différentes méthodes d'AC

La table 6 indique les moyennes des résultats ainsi que l'écart type (entre parenthèses) sur 50 exécutions des différents contrôleurs, ainsi que les exécutions avec les opérateurs Uniforme, FF, CC et CCTM. Les meilleurs résultats sont marqués en gras.

	f500		aim-100		3bitadd		ibm		simon	
Unif	218.4	(5.7)	11.2	(1.0)	5781.0	(141.5)	34149.6	(138.5)	2872.6	(33.9)
FF	30.2	(4.9)	1.9	(0.6)	164.7	(24.5)	3827.8	(160.5)	137.5	(9.7)
CC	7.2	(1.3)	1.9	(0.6)	12.2	(3.2)	1247.7	(98.67)	81.6	(5.4)
CCTM	7.3	(1.4)	1.8	(0.6)	**12.2**	**(2.6)**	1237.2	(78.1)	**81.2**	**(5.3)**
C-M2	25.9	(24.0)	2.4	(1.9)	469.6	(83.7)	6009.7	(3024.6)	189.0	(17.3)
C-M2C	16.8	(19.7)	2.6	(1.9)	519.7	(80.0)	6063.4	(2171.8)	194.4	(23.9)
C-PP	56.3	(33.6)	2.2	(1.9)	481.9	(137.5)	5151.9	(2758.8)	209.2	(41.8)
DP-M2	67.4	(62.7)	3.3	(2.2)	416.5	(453.7)	2712.0	(3523.9)	94.3	(103.0)
DP-M2C	53.3	(59.0)	2.5	(1.5)	266.0	(405.4)	2567.1	(4206.3)	108.0	(102.5)
DP-PP	**6.0**	**(1.4)**	**1.0**	**(0.0)**	303.3	(47.7)	**423.8**	**(75.2)**	93.5	(7.7)
RP-M2	98.2	(53.8)	2.9	(1.7)	534.6	(478.9)	12370.3	(5214.2)	201.0	(188.7)
RP-M2C	104.2	(52.5)	2.6	(1.8)	395.2	(431.3)	12050.3	(5141.1)	277.9	(200.0)
RP-PP	7.8	(1.5)	1.0	(0.0)	517.1	(51.8)	4495.9	(791.9)	148.9	(11.9)
C-A	21.0	(14.4)	1.1	(0.2)	404.9	(59.7)	4908.4	(1623.0)	183.7	(16.7)

TAB. 1 – Nombre de clauses fausses et écart type

Le meilleurs résultats obtenus pour un contrôleur sont ceux de DP-PP, qui améliorent le meilleur des croisements de l'état de l'art dans trois instances. La table 6 montre les pourcentages d'amélioration de DP-PP.

	f500	aim-100	3bitadd	ibm	simon
Unif	**97,23%**	**91,07%**	**94,75%**	**98,76%**	**96,75%**
FF	**80,01%**	**47,92%**	-84,13%	**88,93%**	**31,99%**
CC	**16,57%**	**47,92%**	-2381,67%	**66,03%**	-14,62%
CCTM	**17,03%**	**45,05%**	-2385,74%	**65,74%**	-15,15%

TAB. 2 – Amélioration du contrôleur DP-PP par rapport aux croisements de l'état de l'art.

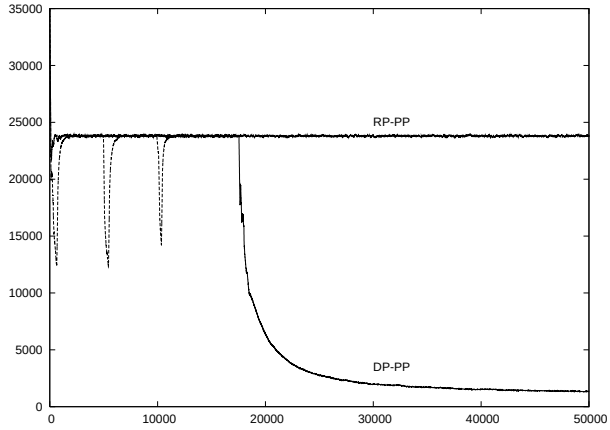

FIG. 9 – Comportement de Rang Pareto versus Dominance Pareto : Nombre moyenne de clauses fausses sur l'instance *ibm*

On peut remarquer que le contrôleur DP-PP produit des résultats comparables à ceux obtenus avec les meilleurs opérateurs sans contrôleur (sauf pour *3bitadd*). Toutefois, rappelons que la mise au point de CC et CCTM (les meilleurs opérateurs sans contrôleur) repose sur un travail de plusieurs semaines de comparaisons, d'analyses et d'expériences [10], tandis que le contrôleur réalise un travail similaire, en quelques minutes et, surtout, sans intervention humaine. De plus, si on considère Notamment l'instance industrielle *ibm*, le nombre de clauses fausses moyenne trouvées avec DP-PP équivaut à $\frac{1}{3}$ de ceux atteints avec CC et CCTM.

En comparant les différentes méthodes d'AC, on peut noter que, en général, DP apparaît dans la plupart des cas comme le contrôleur le plus performant, suivi par C puis RP. On pourrait se demander d'abord quelle est la différence entre les deux contrôleurs fondés sur les notions de la dominance Pareto. Pour répondre à cela il faut étudier leur comportement lors des exécutions individuelles. La figure 9 montre la qualité moyenne de la population en utilisant DP-M2 et RP-M2 sur l'instance *ibm*.

RP considère également tous les opérateurs placés dans la frontière de Pareto (points dans la figure 5.c avec une valeur égale à 0), ce qui induit un équilibre entre les tendances à explorer et à exploiter et empêche l'AE de pencher d'un coté plutôt que de l'autre. Notons que les tentatives d'augmentation de la qualité de RP-PP sont modérées, en provocant le retour de la qualité moyenne vers une valeur intermédiaire. Par contre, si on utilise DP, le fait de mieux considérer les opérateurs qui sont dans la tendance générale (points dans la figure 5.b avec une grande valeur) permet l'abandon de ce *statu quo* et d'améliorer finalement la qualité de la population. Cet "équilibre flexible" est le principal atout de cette méthode d'AC.

En ce qui concerne les méthodes de SO, on trouve un avantage marqué pour PP. M2 semble être également avantageux sur *simon*, mais en général son application ne produit pas de résultats concluants, ce qui est également le cas pour M2C. D'une façon étonnante, A n'est pas décevant (il faut se rappeler que cette SO choisit parmi les opérateurs que le Forgeron autorise à survivre).

Étant donné que PP et A produisent tous deux de bons résultats, on peut en déduire que l'exploration de

M2 et M2C a été trop conservatrice, et qu'on pourrait tirer plus de profits avec un réglage qui ne soit pas aussi focalisé sur l'exploitation. En général, on peut conclure que les aspects les plus importants du contrôleur sont le contrôle des paramètres structuraux (i.e., le choix des opérateurs disponibles pendant la recherche), l'évaluation de ces opérateurs (i.e., la méthode de CA) et finalement leur application (mécanisme de la SO).

7 Conclusion

Dans cet article nous avons proposé un contrôleur permettant de créer et d'appliquer de manière autonome des opérateurs de croisements pour les AEs. Ce contrôleur s'appuie sur la qualité des individus de la population mais prend aussi en compte le contrôle de la diversité. De façon plus générale, tout type de paramètres associés aux probabilités d'application des opérateurs peut être intégrés dans ce contrôleur.

Afin d'évaluer l'apport d'un tel mécanisme, nous l'avons intégré dans un AE travaillant sur le problème SAT. Le contrôleur a été comparé aux meilleurs opérateurs de croisement de l'état de l'art et il s'est avéré très compétitif voire même meilleur dans certains cas. De plus, grâce à l'automatisation du processus de conception, aucune expérience n'a été nécessaire pour choisir les opérateurs à fournir au contrôleur.

Des pistes pour un travail futur sont l'élaboration de schémas plus raffinés pour le contrôle des paramètres structuraux et l'incorporation d'un mécanisme de contrôle de la recherche [13]. Il serait aussi intéressant d'analyser les opérateurs les plus utilisés par DP-PP afin de vérifier l'intérêt du contrôleur en tant qu'outil d'assistance à la conception. Finalement, on peut envisager l'application de ce contrôleur sur d'autres opérateurs et algorithmes, tels que le Tabu Search ou le Simulated annealing).

Références

[1] R. Battiti and M. Brunato. *Learning and Intelligent Optimization. Proc. Int. Conf., LION 2007*, volume 5313 of *LNCS*. Springer, 2008.

[2] R. Battiti, M. Brunato, and F. Mascia. *Reactive Search and Intelligent Optimization*, volume 45 of *Operations Research/Computer Science Interfaces*. Springer Verlag, 2008.

[3] K.A. De Jong and W.M. Spears. Using GA to solve NP-complete problems. In *Proc. of Int. Conf. on GA ICGA*, pages 124–132, 1989.

[4] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Trans.Evol.Computation*, 3 :124–141, 1999.

[5] A. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Extreme value based adaptive operator selection. In G. Rudolph et al., editor, *Proc. PPSN'08*, pages 175–184. Springer, 2008.

[6] C. Fleurent and J.A. Ferland. Object-oriented implementation of heuristic search methods for graph coloring, maximum clique, and satisfiability. In *Cliques, Coloring, and Satisfiability*.

[7] M.R. Garey and D.S. Johnson. *Computers and Intractability , A Guide to the Theory of NP-Completeness*. W.H. Freeman & Company, San Francisco, 1979.

[8] J. Gottlieb and N. Voss. Adaptive fitness functions for the SAT problem. In *Proc. PPSN*, pages 621–630. Springer Verlag, 2000. LNCS 1917.

[9] Y. Hamadi, E. Monfroy, and F. Saubion. Special issue on autonomous search. *Constraint Programming Letters*, 4, 2008.

[10] F. Lardeux, F. Saubion, and J-K Hao. GASAT : A genetic local search algorithm for the sat problem. *Evolutionary Computation*, 14(2) :223–253, 2006.

[11] F. Lobo, C. Lima, and Z. Michalewicz, editors. *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*. Springer, 2007.

[12] J. Maturana, A. Fialho, F. Saubion, M. Schoenauer, and M. Sebag. Compass and dynamic multi-armed bandits for adaptive operator selection. In *Proc. of IEEE Congress on Evolutionary Computation CEC*, 2009. to appear.

[13] J. Maturana and F. Saubion. On the design of adaptive control strategies for evolutionary algorithms. In *Proc. Int. Conf. on Artificial Evolution. LNCS 4926, Springer*, 2007.

[14] J. Maturana and F. Saubion. A compass to guide genetic algorithms. In G. Rudolph et al., editor, *Proc. PPSN'08*, pages 256–265. Springer, 2008.

[15] V. Pareto. Cours d'économie politique. in Vilfredo Pareto, Oeuvres complètes, Genève : Librairie Droz, 1896.

[16] C. Rossi, E. Marchiori, and J.N. Kok. An adaptive evolutionary algorithm for the SAT problem. In *Proc. of Symposium on Applied Computing SAC*, pages 463–470. ACM press, 2000.

[17] L. Sais. *Problème SAT : progrès et défis*. Collection Program. par contraintes. Hermès, 2008.

[18] D. Thierens. *Parameter Setting in Evolutionary Algorithms*, chapter Adaptive Strategies for Operator Allocation, pages 77–90. Volume 54 of Lobo et al. [11], 2007.