Algorithme génétique multi-objectifs adaptatif

Wahabou Abdou[1], Christelle Bloch[2], Damien Charlet[2], François Spies[2]

[1]Laboratoire d'Électronique, Informatique et Image
UMR CNRS 6306 – Université de Bourgogne, Dijon, France
wahabou.abdou@u-bourgogne.fr

[2]Institut FEMTO-ST, Département d'Informatique des Systèmes Complexes
UMR CNRS 6174 – Université de Franche-Comté, Montbéliard, France
{*christelle.bloch, damien.charlet, francois.spies*}*@univ-fcomte.fr*

Résumé— **Cet article présente une version adaptative d'un algorithme génétique multi-objectif appelé GAME. Ce dernier propose une structuration de la population en plusieurs fronts de Pareto et la définition de diverses fonctions de fitness. La version proposée dans cet article, aGAME, introduit un opérateur d'adaptation dynamique qui alterne les phases d'exploration et d'exploitation au cours de l'exécution de l'algorithme. Ces changements de mode de parcours de l'espace de recherche utilisent des indicateurs de performance afin de maintenir un équilibre entre la "convergence" et la diversité des solutions obtenues. aGAME est comparé à la version de base (GAME) ainsi qu'aux trois meilleurs algorithmes de la compétition CEC 2009, sur des problèmes bi-objectifs avec des contraintes. Cette étude comparative montre que aGAME obtient de meilleurs résultats que les quatre algorithmes auxquels il est comparé. Ces résultats valident l'efficacité de l'opérateur d'adaptation dynamique et les performances globales de l'algorithme.**

Mots-clés— **Algorithme génétique multi-objectifs, algorithme adaptatif, fonction de fitness, indicateurs de convergence et diversité, fronts de Pareto.**

I. Introduction

Au cours des dernières décenies, plusieurs travaux traitant des algorithmes évolutionnaires (AE) ont porté sur les aspects multi-objectifs en utilisant des fronts de Pareto et une archive de solutiuons non-dominées. L'adaptativité des algorithmes consitue également un sujet de recherche actif. La section II présente un état de l'art sur ces travaux. Cet article propose un nouvel algortihme génétique multi-objtectifs adaptatif appelé aGAME (*adaptive Genetic Algorithm with Multiple parEto sets*). Il s'agit d'une version améliorée de l'algorithme GAME [1] qui est décrit à la section III. GAME utilise une structuration de la population en plusieurs fronts de Pareto, de nouvelles stratégies de sélection et diverses fonctions de fitness. aGAME introduit un opérateur d'adaptation dynamique afin d'améliorer, à la fois, les performances et la robustesse de GAME. Les spécificités de ce nouvel algorithme sont décrites à la section IV. aGAME alterne entre quatre modes de parcours de l'espace de recherche : initialisation (ou exploration initiale), mode normal, exploration et exploitation. Les changements de modes dépendent de valeurs d'indicateurs de convergence et de diversité. À la section V, aGAME est comparé, dans un premier temps, à sa version statique (GAME) afin de valider l'impact de l'opérateur d'adaptation dynamique. Dans un second temps, le nouvel algorithme proposé est comparé aux trois meilleurs algorithmes de la compétition CEC 2009 [2] sur des problèmes bi-objectifs avec des contraintes.

II. Algoritmes évolutionnaires multi-objectifs

A. Algorithmes utilisant la Pareto dominance

De nombreux algorithmes évolutionnaires multi-objectifs (*Multi-Objective Evolutionary Algorithms, MOEAs*) utilisent la notion de Pareto dominance pour classer les solutions et définir des statégies de sélection (de reproduction ou de survie). En général, ce sont des algorithmes élitistes. Ils gardent les meilleures solutions dans la population au fil des générations ou les sauvegardent dans une archive. Dans le premier cas, ces solutions participent au processus de reproduction et peuvent donc influer le parcours de l'espace de recherche. Toutefois, le nombre de solutions non-dominées peut très rapidement croître pour des problèmes à plusieurs objectifs, reduisant de fait le nombre de places possibles (dans la population) pour de nouvelles solutions. Pour pallier ce problème, un opérateur de maintien de la diversité est souvent utilisé. NSGA-II (*Elitist Non-dominated Sorting Genetic Algorithm*) [3] est l'un des algorithmes les plus connus dans cette catégorie. Dans le cas de l'approche utilisant une archive de solutions non-dominées, ces dernières ne participent pas nécessairement à la reproduction. La notion dominance est souvent, dans ce cas, combinée à d'autres indicateurs pour la définition de valeurs de fitness. Dans cette catégorie, on peut citer SPEA2 (*Strength Pareto Evolutionary Algorithm 2*) [4] qui est un algoritme évolutionnaire basé sur des indicateurs (*Indicator-based evolutionary algorithms*) [5].

NSGA-II utilise outre une stratégie élitiste, un mécanisme de préservation de la diversité. À chaque génération, la population des parents et celle des enfants sont fusionnées et classées en plusieurs fronts de Pareto. La population de la génération suivante est constituée en choisissant des solutions dans ces fronts de Pareto en commençant par le premier. Lorsque la taille du front à utiliser est supérieure au nombre de places restantes à pourvoir dans la future population, les solutions sont choisies selon leur valeur de *crowding distance* [3]. Il s'agit d'un indicateur qui calcule la distance moyenne, sur l'ensemble des objectifs, entre une solution donnée et ses voisins directs dans l'espace des résultats (l'espace des objectifs).

SPEA2 utilise une archive (de taille fixe) de solutions non-dominées. En plus de servir de cache d'élites, cette archive participe aux opérations génétiques et influence donc l'exploration de l'espace de recherche. La fonction de *fitness* est basée sur la notion de dominance et la densité du voisinage. Pour chaque solution $\overrightarrow{x_i}$ la fonction de *fitness* tient

compte aussi bien de la qualité des solutions la dominant que de celle des solutions qu'elle domine.

Plus récemment, FastPGA (*Fast Pareto Genetic Algorithm*) [6] a combiné les idées proposées dans NSGA-II et SPEA2. FastPGA classe les parents et les enfants en deux ensembles : solutions non-dominées et celles qui sont dominées. Pour les solutions non-dominées, la valeur de *fitness* est calculée en utilisant la notion de *crowding distance* introduite dans NSGA-II. Pour les autres solutions, la procédure de calcul de la valeur de *fitness* ressemble à celle utilisée dans SPEA2. En outre, FastPGA utilise un mécanisme lui permettant de réguler dynamiquement la taille de la population. Ce qui lui permet d'avoir de bonnes performances pour la résolution des problèmes ayant des fonctions objectifs complexes et coûteuses en temps de calcul, ou pour des problèmes utilisant des simulateurs.

IBEA (*(Indicator-based Evolutionary Algorithm*) [5] a un processus de sélection qui repose explicitement sur des calculs d'indicateurs ou de métriques. SPEA2 utilise un principe quasi-similaire, puisque le calcul des *fitness* y prend en compte les relations de dominance et la densité. En général, les indicateurs de performance étaient définis pour mesurer les performances des algorithmes multi-objectifs en fin d'exécution. L'une des innovations dans IBEA est de les utiliser pour définir la *fitness* des individus. Dans la lignée de IBEA, de nombreux algorithmes évolutionnaires [7] et approches de recherche locale [8] ont été développés en utilisant majoritairement le critère de l'hypervolume pour définir leur processus de sélection.

B. Adaptive MOEAs

De nombreux travaux de recherche récents portent soit sur des approches statistiques pour régler les paramètres [9] [10] ou sur des algorithmes adaptatifs.

Après avoir fait la différence entre réglage de paramètres et contrôle de paramètres, Eiben et al. [11] subdivisent ce dernier groupe en trois sous-groupes : déterministe, adaptatif et auto-adaptatif.

La catégorie "réglage de paramètres" contient les algorithmes pour lesquels les valeurs des paramètres sont fixés et ne changent pas durant l'exécution des algorithmes. Pour garantir la robustesse de l'algorithme concernant le choix des paramètres et leurs valeurs, une étude paramétrique minutieuse peut être réalisée à l'aide d'outils statistiques [10].

Le contrôle déterministe permet de modifier les paramètres indépendamment des résultats intermédiaires de l'algorithme. Typiquement, des indicateurs tels que le nombre ou le numéro de génération sont utilisés pour faire varier les valeurs de ces paramètres à l'aide d'une liste de valeurs possibles ou d'une fonction déterministe. Par exemple, Saw-Tooth GA (*Saw-Tooth Genetic Algorithm*) [12] fait varier la taille de la population à une fréquence régulière. Tan et al. [13] ont quant à eux proposé une méthode pour les algorithmes génétiques binaires qui associe à chaque bit du chromosome un taux de croisement et un taux de mutation. L'idée étant de favoriser le changement de bits de poids fort au début de l'exécution pour permettre une large exploration. Cette probabilité décroîtra au fil des générations. Mais selon De Jong [14], la prédiction du comportement d'un algorithme à une génération donnée n'étant pas évidente, les méthodes basées sur le contrôle déterministe sont difficilement applicables.

Grâce au monitoring de certaines propriétés de l'algorithme, les méthodes adaptatives modifient le processus d'évolution du programme d'optimisation. Deb et al. [15] ont proposé un opérateur de croisement (*Simulated Binary Crossover, SBX*) qui adapte l'évolution de l'algorithme en fonction des *fitness* des parents et leurs enfants.

Enfin les algorithmes auto-adaptatifs insèrent les informations liées à la stratégie d'adaptation dans le génome des solutions. Ces informations peuvent, par exemple, être les paramètres à réguler. Ces gènes supplémentaires participent aux opérations de reproduction (notamment le croisement). Les solutions ayant de bonnes stratégies d'adaptation survivront et transmettront ces informations à leurs descendants. Hinterding et al. [16] ont introduit la mutation gaussienne qui utilise ce principe. Le paramètre permettant de contrôler la fonction gaussienne est ajouté au chromosome.

L'agorithme proposé dans cet article, aGAME, appartient à la famille des algorithmes adaptatifs : certains de ses paramètres, tels que le taux de mutation et d'autres paramètres de sélection, varient au cours de son exécution. Ces variations dépendent d'indicateurs de performance, qui sont périodiquement calculés pour évaluer la qualité et la diversité des solutions.

III. GAME : un algorithme génétique à multiple fronts de Pareto

GAME (Genetic Algorithm with Multiple parEto sets) structure la population en plusieurs fronts de Pareto comme NSGA-II et propose diverses fonctions de *fitness* à l'instar de FastPGA. GAME utilise également une stratégie de sélection en deux étapes. Tous ces points seront détaillés dans cette section.

A. Classement des solutions et fonctions de fitness

GAME répartit les solutions en plusieurs fronts de Pareto. Cela permet un passage progressif des bonnes solutions (premier front, PF_1) aux moins bonnes (solutions dominées, dernier front). Les solutions non-dominées de chaque génération sont sauvegardées dans une archive (PF_0). Lorsque le problème à traiter comporte des contraintes, celles-ci sont gérées au moment de la construction des fronts de Pareto. GAME utilise une gestion de contraintes basée sur des pénalités. Afin de préserver la primauté des solutions qui respectent les contraintes, celles qui les violent sont interdites du premier front de Pareto. Toutefois, lorsque ce dernier est vide, des solutions qui ne satisfont pas les contraintes peuvent être acceptées. Cette situation est souvent rencontrée au cours des premières générations. Mais elle est assez vite corrigée car les solutions respectant les contraintes auront une plus grande probabilité de reproduction et transmettront donc leurs caractères à leur descendance.

Le calcul de la valeur de *fitness* dépend du front de Pareto auquel appartient la solution. GAME distingue trois formules de *fitness*. Pour les solutions de l'archive et de PF_1, GAME utilise un indicateur de diversité. La *fitness* qui leur est associée est leur valeur de *crowding distance* [3]. Cet indicateur renseigne sur la distribution des solutions dans le front. Pour les soultions du deuxième front (PF_2),

GAME privilégie celles qui sont les plus proches de PF_1. L'indicateur de convergence utilisé pour mesurer la proximité des solutions des deux fronts est la *generational distance*. Enfin, pour les autres fronts, un nouvel indicateur est utilisé pour le calcul des valeurs de *fitness*. Il s'agit du **gain**. Le gain d'une solution $\overrightarrow{x_i}$ par rapport à $\overrightarrow{x_j}$ pour la fonction objectif f_k est noté $gain(\overrightarrow{x_i}, \overrightarrow{x_j}, k)$. Il exprime l'amélioration apportée par $\overrightarrow{x_i}$ par rapport à $\overrightarrow{x_j}$ sur la fonction choisie. Ce gain, défini par l'équation 1, prend des valeurs comprises entre -1 et 1. Une valeur négative indique que la première solution est moins bonne que la seconde sur cet objectif. Un gain nul signifie que les deux solutions sont équivalentes.

$Gain(\overrightarrow{x_i})$, présenté à l'équation 1, exprime l'amélioration, sur l'ensemble des objectifs, apportée $\overrightarrow{x_i}$ par rapport aux autres solutions du même front.

$$gain(\overrightarrow{x_i}, \overrightarrow{x_j}, k) = \frac{\lambda(f_k(\overrightarrow{x_i}) - f_k(\overrightarrow{x_j}))}{Max(f_k(\overrightarrow{x_i}), f_k(\overrightarrow{x_j}))} \quad (1)$$

où λ est un coefficient qui vaut 1 pour un problème de maximisation et -1 pour une minimisation.

La *fitness* de chaque individu est la somme de ses gains (pour l'ensemble des objectifs) par rapport aux autres solutions du même front (voir équation 2).

$$fitness(\overrightarrow{x_i}) = \sum_{j=1}^{|PF|} \sum_{k=1}^{m} gain(\overrightarrow{x_i}, \overrightarrow{x_j}, k) \quad (2)$$

où
- $|PF|$ est la taille du front auquel appartient $\overrightarrow{x_i}$;
- m est le nombre d'objectifs du problème à traiter ;

B. Stratégie de sélection

GAME utilise une sélection en deux étapes. La première consiste à choisir un front PF_i parmis ceux construits. Si une probabilité de sélection élevée est affectée à un front de Pareto de faible taille, les risques d'obtenir des individus (quasi-)identiques à la génération suivante croissent et le niveau de diversité au sein de la population baisse. Pour éviter ce problème, GAME affecte à chaque front PF_i une probabilité d'être sélectionné qui tient compte non seulement de son rang i mais aussi de sa taille. Ainsi l'algorithme s'adapte au nombre de fronts construits et à leurs cardinalités. La seconde étape de sélection permet de choisir une solution dans le front sélectionné. Elle s'appuie sur un tournoi basé sur les valeurs de *fitness* des solutions.

Les paramètres de l'algorithmes ont été réglés à l'aide d'un plan d'expérience et des outils statistiques. Bien que ce réglage ait donné des résultats prometteurs, il n'est certainement pas indépendant du type de problème étudié. Ceci a motivé la conception d'une version adaptative de GAME, appelée aGAME, dans laquelle un nouvel opérateur d'adaptation dynamique est ajouté pour renfocer la robustesse de l'algorithme. La section suivante décrit cette contribution.

IV. Opérateur d'adaptation dynamiqe

L'adaptation dynamique doit permettre aux MOEAs de s'adapter à différents "paysages" de l'espace de recherche, en particulier pour détecter des optima locaux et pouvoir s'en extraire. Elle doit également améliorer les performances de l'algorithme en conduisant aux solutions à la fois très diverses et aussi près que possible de optimum global.

Selon [17], un MOEA efficace doit satisfaire deux buts : explorer largement l'espace de recherche et avancer vers la ou les solutions optimales. De nombreux travaux [13] [18] [19] [20] ont mis en évidence que l'exploration devrait être privilégiée dans les premières générations . Cette approche est connue dans la littérature par l'expression *explore first, exploit later*. Elle a l'avantage de permettre, au début de l'exécution de l'algorithme, un large parcours de l'espace de recherche afin d'identifier les solutions performantes. Dans une deuxième étape, la phase d'exploitation, le voisinage de ces solutions est plus intensément étudié.

Cet article propose une nouvelle approche pour maintenir l'équilibre entre les phases d'exploration et d'exploitation. L'opérateur d'adaptation proposé dynamique alterne périodiquement divers modes de parcours de l'espace de recherche tout au long de l'exécution de aGAME. Les changements de mode se produisent en fonction des valeurs d'indicateurs de qualité et de diversité. Le paragraphe suivant présente les indicateurs choisis et motive ce choix.

A. Indicateurs de performance

Dans la littérature, de nombreux indicateurs ont été proposés pour évaluer la qualité des solutions retournées par des MOEAs. aGAME utilise ces indicateurs pour contrôler le comportement de l'algorithme. L'opérateur d'adaptation proposé dynamique utilise deux indicateurs distincts : l'un pour la convergence l'autre pour la diversité. Ce choix de deux indicateurs plutôt qu'une seule métrique telle que l'hypervolume qui mesure à la fois la diveristé et la convergence a été fait pour assurer l'évolutivité de aGAME. En effet, le calcul de l'indicateur hypervolume est NP-difficile et son utilisation est souvent limitée à un faible nombre d'objectifs. En outre, Deb et Jain [21] explique qu'il est préférable d'utiliser un indicateur pour mesurer la diversité des solutions et un autre pour évaluer la convergence vers le front optimal (PF^*). Par conséquent, l'opérateur proposé, DAO, utilise l'IGD (*Inverted Generational Distance*) pour mesurer la proximité entre les solutions évaluées et PF^*, lorsque ce dernier est connu. Dans le cas contraire, l'archive des solutions non-dominées, qui constitue la meilleure estimation de PF^*, est utilisée comme front de référence. l'IGD repose sur le calcul d'une distance euclidienne pour évaluer la moyenne de la distance minimale entre tous les points de PF^* et ceux de PF (voir l'équation 3).

$$IGD = \frac{\left(\sum_{i=1}^{|PF^*|} d_m^*(\overrightarrow{x_i^*})\right)^{1/m}}{|PF^*|} \quad (3)$$

avec

$$d_m^*(\overrightarrow{x_i^*}) = \min_{j=1}^{|PF|} \sqrt{\sum_{k=1}^{m} (f_k(\overrightarrow{x_i^*}) - f_k(\overrightarrow{x_j}))^2} \quad (4)$$

où $\overrightarrow{x_i^*}$ et $\overrightarrow{x_j}$ sont des solutions appartenant respectivement à PF^* et PF.

DAO utilise aussi le *diversity running performance metric* (DM), un indicateur de diversité proposé par

Deb et Jain [21]. Cet indicateur évalue la distribution des solutions dans un front donné, pour chaque fonction objectif. DM est calculé à chaque génération, en réalisant des projections des solutions non-dominées (celles appartenant à PF_1) dans un hyperplan. Ce dernier est divisé en n hypercubes [1]. Le niveau de diversité dépend de la présence ou de l'absence de solutions dans chaque hypercube. DM est maximal lorsque chaque hypercube contient au moins une solution. Zeng et al. [22] [23] ont proposé de définir n comme étant le rapport de la taille de la population sur le nombre d'objectifs.

DAO calcule périodiquement les valeurs de l'IGD et DM. L'IGD est utilisé pour détecter si aGAME ne parvient pas à réduire la distance entre la population et le front de référence. Le cas échéant, DM vérifie si cela est dû à un manque de diversité, ce qui pourrait indiquer que l'algorithme est piégé dans un optimum local. Selon ces éléments, aGAME choisit le meilleur mode de parcours de l'espace de recherche.

B. Changements de modes

Le DAO utilise le principe *explore first, exploit later*. Il distingue une phase d'**exploration initiale** au cours de laquelle l'algorithme découvre assez largement l'espace de recherche. Elle a une durée t_{init} exprimée en nombre de générations. Durant cette phase, la probabilité de mutation (p_m) dépend du niveau de diversité au sein de la population (voir équation 5).

$$p_m = 1 - DM(population) \tag{5}$$

$DM(population)$ prend des valeurs comprises entre 0 et 1. Plus elles sont proches de 1, meilleur est le niveau de diversité au sein de la population. Lorsque $DM(population)$ tend vers 0, traduisant une population peu diversifiée, p_m croît afin de favoriser l'opération de mutation.

Au delà de t_{init}, il est nécessaire d'accentuer l'exploitation tout en prévenant d'éventuels pièges dans des optima locaux. Le DAO surveille régulièrement (avec une périodicité $t_{monitoring}$) le déroulement de l'algorithme et réoriente son évolution lorsque cela est nécessaire. La figure 1 présente le fonctionnement général de DAO.

Lorsque l'IGD de l'archive des solutions n'a pas varié entre deux monitorings consécutifs, alors une phase de stagnation de l'IGD a commencé [2]. La correction à apporter dépend du niveau de diversité au sein des meilleures solutions évaluées à l'instant t. Si elles sont plus diversifiées que celles de la génération précédente, alors il n'y a certainement pas de perte de diversité dans la population courante. Le comportement à adopter, **mode exploitation**, consiste à intensifier la recherche de nouvelles solutions autour des meilleures solutions connues. L'archive des solutions non-dominées participera donc aux opérations de reproduction. Lors de la sélection des parents, elle aura une probabilité d'être choisie plus élevée que tout autre ensemble de solutions.

1: t : numéro de la génération
2: t_{init} : durée de la phase d'initialisation
3: $t_{monitoring}$: période de monitoring de l'algorithme
4: t_d : numéro de la génération au cours de laquelle a eu lieu le dernier monitoring
5: $P(t)$: population à la génération t
6: $PF_1(t)$: premier front de Pareto à la génération t
7: $A(t)$: archive des solutions non-dominées construite jusqu'à la génération t
8: *mode* : comportement que devra adopter l'algorithme (*normal*, *exploration*, *exploration initiale*, ou *exploitation*)
9: **if** $t < t_{init}$ **then**
10: $mode \leftarrow exploratio\ initiale$
11: **else**
12: **if** $t\%t_{monitoring} = 0$ **then**
13: **if** $IGD(A(t)) = IGD(A(t_d))$ **then**
14: **if** $DM(PF_1(t)) > DM(PF_1(t-1))$ **then**
15: $mode \leftarrow exploitation$
16: **else**
17: $mode \leftarrow exploration$
18: **end if**
19: **else**
20: $mode \leftarrow normal$
21: **end if**
22: **end if**
23: $t_d \leftarrow t$
24: **end if**
25: return *mode*

Fig. 1. Algorithme d'adaptation dynamique de l'exploration et de l'exploitation

Si on observe une constance des valeurs des IGD et que la diversité au sein des solutions du premier front de Pareto (PF_1) est stable ou en baisse, il se pourrait que l'algorithme soit dans un optimum local ou global. S'il s'agit de l'optimum global, il est normal de ne plus observer d'amélioration de la qualité des solutions proposées. Toutes les tentatives de correction par le DAO seront improductives, sans toutefois dégrader la qualité de l'archive. Si au contraire l'algorithme se trouve dans un optimum local, augmenter la probabilité de mutation pourrait permettre à l'algorithme de s'extirper de ce piège. La stratégie à utiliser est alors le **mode exploration**. L'équation 6 précise la variation de la probabilité de mutation.

$$p_m = DM(PF_1) \tag{6}$$

Si le DAO ne constate pas de stagnation de l'IGD lors d'un monitoring, il sélectionne le **mode normal** qui a un fonctionnement semblable à celui de GAME. Dans ce cas, la probabilité de mutation prend une valeur habituellement utilisée par des algorithmes génétiques (équation 7).

$$p_m = \frac{1}{nombre\ de\ variables\ de\ décision} \tag{7}$$

Ainsi, une supervision régulière basée sur le DAO permet de guider aGAME tout au long de son exécution.

1. Hypercubes de dimension $m-1$; m étant le nombre de fonctions objectifs

2. Le calcul de l'IGD de l'archive suppose que la ou les solutions optimales sont connues. Si elles ne sont pas disponibles, l'archive peut être considérée comme front de référence et l'instruction **Si** $IGD(A(t)) = IGD(A(t_d))$ sera remplacée par **Si** $IGD(PF_1(t)) \leq IGD(PF_1(t_d))$ dans l'algorithme de la figure 1

V. Validation et évaluation des performances

Les spécificités de aGAME sont validées dans cette section. L'impact des multiples fronts de Pareto a été présenté dans [1]. Cet article s'interessera à l'influence de l'operateur d'adaptation dynamique. aGAME sera comparé à GAME ainsi qu'à d'autres algorithmes de la littérature.

A. Procédure experimentale

Les résultats présentés dans cet article portent sur les problèmes bi-objectifs avec contraintes proposés pour la compétition CEC 2009 [2]. Il s'agit de sept problèmes de minimisation ayant chacun dix variables de décision.

Conformément à la procédure de tests de la compétition CEC 2009, chaque algorithme est exécuté trente fois pour chaque problème. Le nombre maximum d'évaluations est fixé à 30 000. GAME et aGAME utilisent un croisement 2-points et une mutation bitflip. Le taux de croisement est de 0,8 et le taux de mutation est $\frac{1}{nombre\ de\ variables\ de\ décision}$ (cette valeur peut être modifiée par le DAO).

L'indicateur de performance utilisé pour comparer les résultats est l'IGD (voir équation 3). Le but est d'obtenir la plus petite valeur possible de l'IGD.

B. Influence de l'opérateur d'adaptation dynamique

Les algorithmes évolutionnaires étant des processus stochastiques, le premier problème avec contraintes de la compétition CEC 2009 a été répété trente fois par GAME aGAME, afin d'évaluer l'impact de l'opérateur adaptation dynamique proposée. L'évolution de l'IGD au fil des générations pour les deux algorithmes est présentée à la figure 2. Sur cette figure, chaque point représente la moyenne des IGDs des trente éxécutions pour un numéro de génération donné.

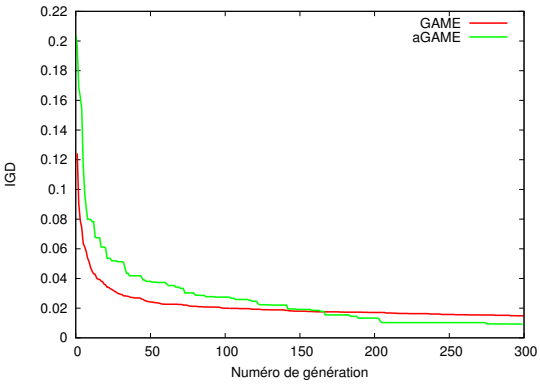

Fig. 2. IGD de GAME et aGAME pour 30 exécutions du premier problème avec contraintes de la compétition CEC 2009

À la fin de l'exécution des algorithmes, aGAME fournit un IGD plus faible que celui de GAME. Toute fois les valeurs de l'IGD de aGAME baisse plus lentement. Cela est dû à la phase d'exploration initiale qui favorise un parcours plus large de l'espace de recherche. Les deux courbes de la figure 2 sont obtenues en calculant la moyenne de l'IGD (de chaque génération) pendant trente exécutions. Un examen détaillé des résultats de GAME pour des exécutions uniques montre que la courbe de l'IGD se présente sous forme d'escaliers, avec des périodes relativement longues de

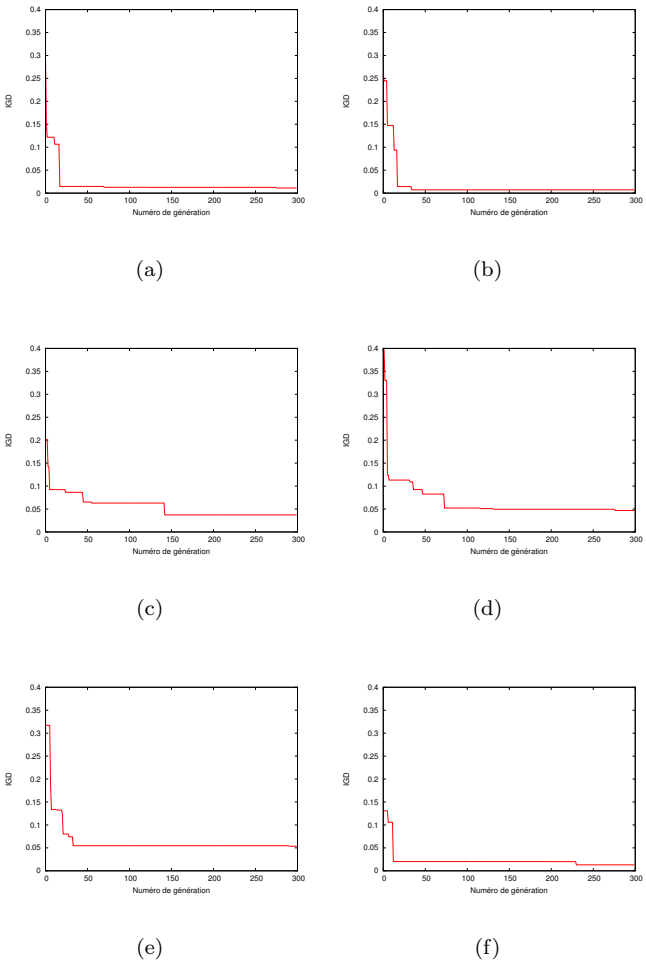

Fig. 3. Courbes de l'IGD pour six exécutions de GAME pour le premier problème avec contraintes de la compétition CEC 2009

stagnation de l'IGD. Cette observation est illustrée par les figures 3 (a) à (f), qui sont représentatives de l'évolution de l'IGD au cours de 6 essais indépendants (avec des 5 fronts de Pareto).

L'expérience qui a permis de tracer la courbe de la figure 3 (a) a été prise avec aGAME (en utilisant la même série de nombres aléatoires) afin d'illustrer l'influence de la DAO sur le déroulement de aGAME . Les valeurs suivantes ont été utilisé pour le DAO :

- $t_{init} = 15\%$ du nombre total de générations
- $t_{monitoring} = 5\%$ du nombre total de générations

L'IGD de l'archive des solutions non-dominées de aGAME (figure 4) ne baisse pas aussi rapidement que dans le cas de GAME (figure 3 (a)). En effet, aGAME commence par une phase d'exploration initiale au cours de laquelle les probabilités de mutations sont généralement plus élevées que dans la version statique. Le but est de permettre un large parcours de l'espace de recherche durant les premières générations et de recueillir le plus d'informations possibles. Les meilleures solutions étant enregistrées dans l'archive, les informations qu'elles contiennent pourront servir lors de la phase d'exploitation.

Pour cet exemple, la phase d'exploration initiale dure 45 générations. Après l'expiration de t_{init}, aGAME alterne entre les modes exploration, normal et exploitation (voir

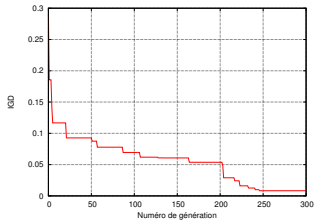

Fig. 4. Amélioration des réultats de la figure 3(a) en utilisant le DAO

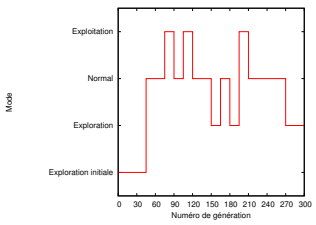

Fig. 5. Changement dynamique de mode pour les résultats de la figure 4

figure 5). Ces changements dynamiques de stratégies permettent à aGAME d'améliorer régulièrement, ou au moins après chaque période de monitoring, son niveau de convergence. Cela permet à aGAME d'obtenir un IGD plus faible à la fin de son exécution.

C. Comparaison avec d'autres algorithmes de la littérature

aGAME est comparé aux trois meilleurs algorithmes de la catégorie "problèmes avec contraintes" de la compétition CEC 2009 : DMOEA-DD [24], LiuLi [25] et MTS [26].

DMOEAD-DD (Dynamical Multiobjective Evolutionary Algorithm - Domain Decomposition) est une amélioration de DMOEA [27] qui utilisait une fonction de *fitness* agrégée incluant la notion de dominance (de Pareto), l'entropie et la densité (sur la base de la *crowding distance*). Dans la nouvelle version, les auteurs utilisent une technique de décomposition pour répartir l'espace de recherche en plusieurs sous-domaines. Chacun d'eux utilise DMOEA pour le calcul des fronts de Pareto. Les différents sous-domaines peuvent s'échanger des informations à l'aide d'opérateurs génétiques.

LiuLi (concaténation des noms des auteurs : Hai-Lin Liu et Xueqiang Li) découpe l'espace de recherche en sous-régions afin de réduire la complexité de l'algorithme. Les opérations génétiques, notamment la reproduction, se font au sein d'une même sous-région. Les échanges d'informations inter-régionaux se font par l'intermédiaire de leurs enfants qui peuvent être affectés à d'autres régions.

MTS (Multiple Trajectory Search) est un algorithme basé sur trois méthodes de recherche locale. Pour chaque solution, MTS détermine la méthode qui correspond à son voisinage. Cet algorithme commence par une recherche large. La taille du voisinage est progressivement réduite jusqu'à atteindre une taille minimale fixée. Ensuite la taille du voisinage reprend sa valeur initiale et sa régression recommence.

GAME est ajouté à l'étude comparative afin d'observer les gains qu'apporte le DAO.

Pour chaque algorithme, le tableau I présente une moyenne des IGD pour 30 éxécutions. Ces résultats montrent que, globalement, GAME présente des faibles de l'IGD et que aGAME permet d'améliorer les résultats de GAME.

VI. Conclusion

En somme, GAME est un algorithme génétique élitiste multi-objectif qui structure la population en plusieurs fronts de Pareto. Ce classement des solutions, associé à une stratégie de sélection en deux étapes a permis d'obtenir des gains à la fois en termes de proximité avec des solutions optimales et en termes de diversité dans l'ensemble des solutions obtenues.

aGAME améliore les résultats de GAME grâce à un nouvel opérateur d'adaptation dynamique. Ce dernier permet à l'algorithme d'alterner les phases d'exploration et d'exploitation afin de s'affranchir d'éventuels optima locaux.

Enfin, en utilisant les conditions expérimentales de la compétition CEC 2009, cet article a montré que aGAME aurait été très bien classé dans cette compétition, ce qui constitue un résultat prometteur. En outre, aGAME utilise une procédure d'évaluation parallélisée selon le un modèle maître-esclave. Ce modèle a permis de réduire la durée d'expérimentation dans des travaux antérieurs dans le domaine des réseaux mobiles [28]. Dans ce contexte, une version précédente de l'algorithme a permis de résoudre des problèmes contraints de quatre objectifs, dont l'évaluation a été effectuée à l'aide d'un simulateur de réseaux. Néanmoins, la conception d'une version asynchrone de aGAME serait très intéressante. Une telle version permetrait au processus maître de générer de nouvelles solutions sans nécessairement attendre que tous les processus esclaves aient terminé leur évaluation.

Références

[1] W. Abdou, C. Bloch, D. Charlet, and F. Spies. Multi-pareto-ranking evolutionary algorithm. In *12th European Conf. on Evolutionary Comp. in Comb. Optimisation (EvoCOP 2012)*, volume 7245, pages 194–205, Malaga, Spain, 2012. Springer.

[2] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari. Multiobjective optimization test instances for the cec 2009 special session and competition. Technical report, The School of Computer Science and Electronic Engineering, 2009.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast Elitist Multi-Objective Genetic Algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6 :182–197, 2000.

[4] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2 : Improving the strength pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology, Zurich, Switzerland, 2001.

[5] E. Zitzler and S. Künzli. Indicator-Based Selection in Multiobjective Search. In *Conference on Parallel Problem Solving from Nature (PPSN VIII)*, pages 832–842. Springer, 2004.

[6] H. Eskandari, C. D. Geiger, and G. B. Lamont. FastPGA : A dynamic population sizing approach for solving expensive multiobjective optimization problems. In *Evolutionary Multi-Criterion Optimization. 4th Int. Conf., EMO 2007*, volume 4403 of *Lecture Notes in Computer Science*, pages 141–155. Springer, 2007.

[7] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang. Multiobjective evolutionary algorithms : A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1) :32 – 49, 2011.

[8] M. Basseur, R.-Q. Zeng, and J.-K. Hao. Hypervolume-based multi-objective local search. *Neural Computing & Applications*, pages 1–13, 2011.

[9] T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors. *Experimental Methods for the Analysis of Optimization Algorithms*. Springer, Germany, November 2010.

[10] J.-L. Hippolyte, C. Bloch, P. Chatonnay, C. Espanet, D. Chamagne, and G. Wimmer. Tuning an evolutionary algorithm with taguchi methods. In *CSTST'08, 5th Int. Conf. on Soft Computing as Transdisciplinary Science and Technology*, pages 265–272. ACM Press, 2008.

[11] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transations on Evolutionary Computation*, 3(2) :124–141, July 1999.

[12] V. K. Koumousis and C. P. Katsaras. A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. *IEEE Trans. Evolutionary Computation*, 10(1) :19–28, 2006.

Problèmes	DMOEADD	LiuLi	MTS	GAME	aGAME
CF 1	0,01131	0,00085	0,01918	0,01489	0,00828
CF 2	0,0021	0,004203	0,02677	0,00042	0,00039
CF 3	0,056305	0,182905	0,10446	0,03462	0,03385
CF 4	0,00699	0,014232	0,01109	0,00742	0,00700
CF 5	0,01577	0,10973	0,02077	0,01227	0,01129
CF 6	0,01502	0,013948	0,01616	0,00181	0,00138
CF 7	0,01905	0,10446	0,02469	0,00545	0,00373
Rang	$3^{ème}$	$4^{ème}$	$5^{ème}$	$2^{ème}$	1^{er}

TABLE I

Valeur moyenne des IGDs de aGAME, GAME et des trois meilleurs algorithmes de la compétition CEC 2009 pour des problèmes avec contraintes

[13] K. C. Tan, S. C. Chiam, A. Al Mamun, and C. K. Goh. Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization. *European Journal of Operational Research*, 197(2) :701–713, 2009.

[14] K. De Jong. Parameter Setting in EAs : a 30 Year Perspective. pages 1–18. 2007.

[15] K. Deb, K. Sindhya, and T. Okabe. Self-adaptive simulated binary crossover for real-parameter optimization. In *Proceedings of the 9th annual conf. on Genetic and evolutionary computation*, GECCO '07, pages 1187–1194, New York, USA, 2007. ACM.

[16] R. Hinterding, Z. Michalewicz, and T. C. Peachey. Self-adaptive genetic algorithm for numeric functions. In *Proceedings of the 4th Int. Conf. on Parallel Problem Solving from Nature*, PPSN IV, pages 420–429, London, UK, 1996. Springer-Verlag.

[17] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley-Interscience Series in Systems and Optimization. John Wiley & Sons, Chichester, 2001.

[18] K. C. Tan, C. K. Goh, Y. J. Yang, and T. H. Lee. Evolving better population distribution and exploration in evolutionary multi-objective optimization. *European Journal of Operational Research*, 171(2) :463–495, 2006.

[19] K. C. Tan, T. H. Lee, and E. F. Khor. Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization. *IEEE Trans. Evolutionary Computation*, 5(6) :565–588, 2001.

[20] N. K. Bambha, S. S. Bhattacharyya, J. Teich, and E. Zitzler. Systematic integration of parameterized local search into evolutionary algorithms. *IEEE Trans. Evolutionary Computation*, 8(2) :137–155, 2004.

[21] K. Deb and S. Jain. Running Performance Metrics for Evolutionary Multi-Objective Optimization. In *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)*, volume 1, pages 13–20, Orchid Country Club, Singapore, November 2002. Nanyang Technical University.

[22] F. Zeng, J. Decraene, M. Y. H. Low, S. Zhou, and W. Cai. Diversity-driven self-adaptation in evolutionary algorithms. 70 :95–106, 2011.

[23] F. Zeng, M. Y. H. Low, J. Decraene, S. Zhou, and W. Cai. Self-adaptive mechanism for multi-objective evolutionary algorithms. In *Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS 2010)*, pages 7–12. Newswood Limited, March 2010.

[24] M. Liu, X. Zou, Y. Chen, and Z. Wu. Performance assessment of dmoea-dd with cec 2009 moea competition test instances. In *IEEE Congress on Evolutionary Comp.*, pages 2913–2918, 2009.

[25] H.-L. Liu and X. Li. The multiobjective evolutionary algorithm based on determined weight and sub-regional search. In *Proceedings of the Eleventh conf. on Congress on Evol. Comp.*, CEC'09, pages 1928–1934, Piscataway, USA, 2009. IEEE Press.

[26] L.-Y. Tseng and C. Chen. Multiple trajectory search for unconstrained/constrained multi-objective optimization. In *Proceedings of the Eleventh conf. on Congress on Evol. Comp.*, CEC'09, pages 1951–1958, Piscataway USA, 2009. IEEE Press.

[27] X. Zou, Y. Chen, M. Liu, and L. Kang. A new evolutionary algorithm for solving many-objective optimization problems. *IEEE trans. on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 38(5) :1402–1412, October 2008.

[28] W. Abdou, A. Henriet, C. Bloch, D. Dhoutaut, D. Charlet, and F. Spies. Using an evolutionary algorithm to optimize the broadcasting methods in mobile ad hoc networks. *Journal of Network and Computer Applications*, 34(6) :1794 – 1804, 2011.