Université Mohamed Khider Biskra

Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie

Département d'informatique

Thèse

Présentée pour l'obtention du diplôme de

docteur en sciences

par

Abdelhakim Cheriet

Métaheuristique hybride pour l'optimisation multi-objectif

Spécialité : Informatique

Soutenue le 11 juillet 2016 devant un jury composé de :

Président du jury	**Pr. Noureddine Djedi**	Université de Biskra
Rapporteur	**Pr. Foudil Cherif**	Université de Biskra
Examinateur	**Pr. Allaoua Chaoui**	Université de Constantine 2
Examinateur	**Pr. Salim Chikhi**	Université de Constantine 2
Examinateur	**Dr. Elkamel Merah**	Université de Kenchela
Examinateur	**Dr. Salim Bitam**	Université de Biskra

Résumé

Dans la plupart des problèmes réels que ce soit technologiques, économiques ou autres, une compétence du point de vue choix de décisions à faire est indispensable. Un décideur doit avoir une très bonne connaissance autour du problème traité pour que les choix puissent être optimaux des points de vue objectifs à atteindre. Néanmoins, les problèmes du monde réel comportent plusieurs objectifs à atteindre, la recherche des solutions qui optimisent les objectifs simultanément rentre dans le domaine d'optimisation multiobjectif. La résolution des problèmes d'optimisation multiobjectif vise à trouver un ensemble de solutions appelé Solution Pareto. Les solutions Pareto sont celles qui ne sont dominées par aucune autre solution. Plusieurs méthodes ont été utilisées pour résoudre ce problème, dont les algorithmes évolutionnaires sont les mieux adaptés. Ceci est du à leurs capacités de trouver les bonnes approximations des solutions Pareto. La plupart des algorithmes évolutionnaires sont classés selon la manière d'intervention du décideur en trois catégories : a priori, a posteriori et interactif. L'algorithme fournit des solutions optimales au décideur qui va choisir celles qu'il le convient. Dans toutes les catégories, si le décideur n'est pas satisfait par les solutions fournies, l'algorithme doit être ré-exécuté, ce qui est vu comme un inconvénient. Afin de faire face à ce problème, l'algorithme utilisé doit être capable de modéliser les solutions obtenues lors de la phase d'optimisation. Un tel modèle sera exploité dans le cas où de nouvelles solutions sont requises. Pour cela, un algorithme à estimation de distribution est utilisé. Il vise à estimer la distribution de meilleures solutions qui sera utilisée pour générer des nouvelles solutions avec les mêmes caractéristiques d'optimalité. Dans cette thèse, un algorithme, nommé CEDA, utilisant les copules pour modéliser les dépendances entre les variables du problème est proposé. Pour bien exploiter les capacités de cet algorithme, une hybridation avec une méthode d'apprentissage SVM est proposée. L'objectif est de minimiser le temps d'exécution dans la phase de mise à jour des solutions surtout dans les problèmes Many-objective. L'utilisation de CEDA sur plusieurs problèmes benchmarks de la littérature montre que notre proposition donne des meilleures qualités.

Mots-clé : Optimisation multiobjectif, métaheuristique, algorithme évolutionnaire, algorithme à estimation de distribution, Copule, Many-objective, SVM.

ملخص

في جميع مشاكل العالم الحقيقي التكنولوجية منها الاقتصادية او غيرها، تحتاج جميعا الى مهارة في اتخاذ القرار. ولذلك يجب ان يكون لصانع القرار معارف جيدة في المجال الذي يواجهه من اجل ان يكون الخيار مثالي من حيث الاهداف المراد تحسينها. إلا ان المشاكل في العالم الحقيقي تتكون من عدة اهداف، البحث عن الحلول التي تحسن كل الاهداف معا يدخل في مجال التحسين المتعدد الاهداف. التحسين المتعدد الاهداف يطمح لايجاد مجموعة من الحلول تسمى حلول باريتو. حلول باريتو هم الحلول التي ليست مسيطر عليهامن اي حلول اخرى. هناك العديد من الطرق التي ترمي الى حل هدا النوع من المشاكل و لقد وجد ان الخوارزميات التطورية والتي هي عبارة عن فئة من الادلة العليا هم الاحسن تكيفا و ذلك لقدرتها على ايجاد تقريبات جيدة لحلول باريتو. معظم الخوارزميات التطورية مقسمت على حسب طريقة تدخل صانع القرار الى قبلية و بعدية و تفاعلية. الخوارزم يقدم الحلول المثلى لصانع القرار و هذا الاخير يختار بين هاته الحلول التي تناسبه. في كل فئات الخوارزميات هذه، اذا كان صانع القرار غير راض على الحلول المقدمة له فانه يجب اعادة تنفيذ الخوارزم من البداية مع امل ايجاد حلول جديدة التي قد ترضي صانع القرار. نقطة الضعف هذه قادتنا للتفكير في ايجاد خوارزم يقوم باعطاء حلول بديلة دون اللجوء الى اعادة تنفيذ الخوارزم من جديد. كان يتوجب على هذا الخوارزم ان يكون له القدرة على نمذجة الحلول التي وجدت في مرحلة التحسين ثم استخداء هذا النموذج لانتاج حلول اخرى دون اعادة تنفيذ مرحلة التحسين مرة اخرى. من اجل انشاء هذا النموذج استخدمنا خوارزم تقدير التوزيع و التي هي فئة من الخوارزميات التطورية. خوارزم تقدير التوزيع هو خوارزم يرمي الى تقدير توزيع الحلول المثلى ثم بعد ذلك استخدام هذا التوزيع من اجل انتاج حلول تكون لها نفس خاصية الامثلية. تقدير التوزيع يكون باستخدام طرق في الاحتمالات و الاحصاء، في اطروحتنا الخوارزم ر. خ. ت. ت استخدمنا فيه الروابط من اجل نمذجة العلاقات بين متغيرات المشكل. و من اجل استفادة اكبر من قدرة الخوارزم اقترحنا استخدام خوارزم مهجن مع طريقة تعلم اوتوماتيكية س. ف. م من اجل التقليص اكثر من وقت التنفيذ في مرحلة تحيين الحلول و خاصة في المشاكل كثيرة الاهداف. ان استخدام رختت على مجموعة من المشاكل النوعية المستخدمة في هذا النوع من الاشكاليات اثبت ان اقتراحنا اعطى نتائج جيدة.

الكلمات المفتاحية: المشاكل المتعدد الاهداف، الخارزميات التطورية، خوارزميات تقدير التوزيع، س. ف. م ، التعلم الاوتوماتيكي، الروابط، مشاكل كثيرة الاهداف.

Abstract

In most real problems, technological, economic or other a competency from the standpoint of choice of decisions to make is essential. A decision maker must have a very good knowledge about the problem addressed so that choices can be optimal from the perspective objectives. However, real world problems has multiple objectives, seeking solutions that optimize objectives simultaneously falls within the field of multi-objective optimization. The resolution of multi-objective optimization problems aim to find a set of solutions called Pareto solution. The Pareto solutions are those that are not dominated by any other solution. Several methods have been used to solve this problem, the evolutionary algorithms are the best suited. This is due to their ability to find good approximations of Pareto solutions. Most evolutionary algorithms are classified according to the manner of intervention of the decision maker into three categories : a priori, posteriori and interactive. The algorithm provides optimum solutions to maker who will choose the ones appropriate. In all classes, if the decision maker is not satisfied with the solutions provided, the algorithm needs to be re-run, which is seen as a disadvantage. To cope with this problem, the algorithm should be able to model the solutions obtained during the optimization phase. Such a model will be operated if new solutions are required. For this, an estimation of distribution algorithm is used. It aims to estimate the distribution of the best solutions that will be used to generate new solutions with the same characteristics of optimality. In this thesis, an algorithm called CEDA, using the copula to model the dependencies between the variables of the problem is proposed. To fully exploit the capabilities of this algorithm, hybridization with a learning method SVM is proposed. The objective is to minimize the execution time in the update solutions phase especially in Many-objective problems. Use of CEDA on several literature benchmarks problems shows that our proposal provides the best qualities.

Keywords : Multiobjective optimization, metaheuristics, evolutionary algorithm, estimation of distribution algorithm, Copula, Many-objective, SVM.

Remerciements

Au terme de ce travail, je tiens à remercier Dieu le tout puissant de m'avoir donné le courage, la volonté et la patience pour achever ce travail.

J'ai l'honneur et le plaisir de présenter ma profonde gratitude et mes sincères remerciements à mon encadreur Pr. Cherif Foudil, pour ses précieuses aides, ces orientations et le temps qu'il m'a accordé pour mon encadrement.

Je remercie profondément tous les enseignants qui m'ont encouragé et soutenu pendant mon cursus.

Un très grand merci à Monsieur Khaled Djaber, Bachir Abdelmalik, Okba Tibermacine, Salim Bitam pour leurs aides, leurs conseils et pour leurs complicités.

Je remercie également les jurys Pr. Noureddine Djedi, Pr. Allaoua Chaoui, Pr. Salim Chikhi, Dr. Elkamel Merah et Dr. Salim Bitam, qui ont accepté de juger mon travail.

Je remercie aussi tous ceux qui ont contribué de prêt ou de loin à la réalisation de mon mémoire.

«« Merci »»

Dédicace

Á Mes Parents.
Á Ma Femme Et Enfants Maram Et Aissa.
Á Mes Sœurs.

Table des matières

Table des figures

Liste des tableaux

Introduction

Les problèmes du monde réel ne comportent pas un seul objectif à optimiser mais souvent un ensemble d'objectifs. Par exemple, dans le domaine de transport, un problème de routage de véhicules (VRP : Vehicle Routing Problem) vise à trouver un nombre minimal de véhicules qui doivent parcourir une distance minimale pour donner un service à un nombre de clients.

Il est clair que les deux objectifs du problème VRP sont contradictoires ; si on veut minimiser le nombre de véhicules utilisés, la distance parcourue augmente et si on minimise la distance parcourue par les véhicules leur nombre augmente. Par conséquent, pour trouver une solution à ce problème, il faut utiliser une relation de différenciation entre les solutions et prendre les meilleures parmi celles réalisables (une solution réalisable est une solution qui satisfait les contraintes du problème).

Comme tout problème multiobjectif, trouver une solution repose sur l'existence d'une solution qui optimise simultanément les différents objectifs du problème. Dans le domaine d'optimisation multiobjectif, la relation de comparaison entre les solutions est appelée relation de *Dominance de Pareto*. Cette relation permet d'évaluer une solution et de conclure si elle est médiocre ou non. Une solution est qualifiée comme non bonne, s'il existe une autre dans l'espace réalisable qui est meilleure que cette solution pour tous les objectifs. Revenons à notre exemple de VRP, une configuration x est meilleure qu'une autre y si x utilise moins de véhicules et ses véhicules parcourent moins de distances. Dans ce cas, on dit que x domine y [39].

L'exploitation de la relation de dominance de Pareto par les méthodes d'optimisation multiobectif a donnée de meilleurs résultats par rapport aux autres méthodes exactes qui n'utilisent pas ce concept. Une méthode exacte est celle qui utilise une résolution exacte pour trouver une solution à un problème d'optimisation multiobjectif. Par exemple, la méthode de la somme pondérée [76] vise à transformer un problème multiobjectif vers un problème mono-objectif en utilisant la somme des objectifs avec des poids d'importance associés à chaque objectif. Il est clair que cette méthode trouve une solution pour

un problème multiobjectif donné qui est une solution exacte mais ce type de méthodes soufre de plusieurs problèmes. Si on prend un problème non-convexe, on va découvrir que la méthode de la somme pondérée va perdre beaucoup de solutions qui peuvent être très utiles dans le processus d'aide à la décision. Cet inconvénient a été surpassé par l'utilisation des méthodes d'optimisation qui utilisent les métaheuristiques.

Dans le domaine de l'optimisation multiobjectif, les métaheuristiques les plus utilisées sont les algorithmes évolutionnaires. L'utilisation des algorithmes évolutionnaires dans l'optimisation multiobjectif rentre dans ce qu'on l'appelle EMO (Evolutionary Multiobjective Optimisation). Un algorithme évolutionnaire à deux principales étapes : la sélection et la reproduction. La sélection est le processus de choisir parmi les solutions (individus) existantes un ensemble de solutions meilleures. Il est à noter que l'adjectif meilleur diffère d'un problème à un autre ; pour un problème multiobjectif les meilleures solutions qu'il faut les choisir sont les solutions qui ne sont dominées par aucunes autres solutions. Dans la littérature, chaque algorithme évolutionnaire utilise son propre méthode de sélection afin de minimiser la complexité de comparaison entre les solutions. La deuxième étape d'un algorithme évolutionnaire est la reproduction. Elle vise à utiliser les solutions sélectionnées dans l'étape de sélection pour générer de nouveaux individus (solutions). Les méthodes de reproduction les plus connues sont le croisement et la mutation utilisées par les algorithmes génétiques. Les algorithmes évolutionnaires ont montré une grande capacité de résolution des problèmes d'optimisation multiobjectifs, soit les Benchmarks utilisée dans la littérature ou dans l'aide à la décision où un Décideur (Decision Maker, DM) doit avoir un ensemble de meilleurs solutions d'un problème donnée pour choisir entre eux.

Selon la manière d'intervention du Décideur (DM) dans le processus de résolution d'un problème, on distingue entre trois classes de méthodes de résolution. Dans les méthodes apriori, où le DM intervient avant le processus de résolution, il doit avoir des connaissances préalables sur le problème traité afin de donner ses préférences pour guider le processus de recherche. Si les solutions obtenues à la fin de la résolution ne le conviennent pas, il doit alors reprendre tout le processus à nouveau. Dans les méthodes interactives, le DM intervient tout au long du processus de résolution ; ce qui prend un énorme temps de résolution. Enfin, si le DM intervient à la fin du processus de résolution, on parle de méthodes a posteriori. Dans ce cas, le DM n'est pas censé avoir des connaissances préalables sur le problème et ne fournit aucune information à la méthode de résolution. Par conséquent, on tombe sur le même problème des méthodes apriori qu'est la non-satisfaction du DM par les solutions obtenues.

Durant les dernières années, d'autres algorithmes évolutionnaires, dits Algorithmes à Estimation de Distribution (Estimation of Distribution Algorithm, EDA) [65], ont été utilisé pour résoudre plusieurs problèmes d'optimisation. Ces algorithmes utilisent des représentations explicites des individus via des lois de probabilités et des modèles statistiques. Un algorithme à estimation de distribution diffère des autres algorithmes évolutionnaires dans la phase de reproduction. En particulier, un EDA commence par la création d'un modèle d'estimation de distribution des meilleurs individus sélectionnés et ensuite génère de nouveaux individus en utilisant ce modèle. Les nouvelles solutions (individus) générées contiennent les caractéristiques de solutions meilleures avec une forte chance qu'elles seront elles aussi meilleures.

Plusieurs types des EDAs ont été utilisé pour résoudre des problèmes mono-objectif et pour le domaine d'optimisation multiobjectif ont fait aussi l'objet de quelque travaux. Le principe de leur utilisation consiste à les adapter pour le cas multiobjecitf. Les EDAs dans l'optimisation multiobjectif reste le même avec un petit changement ; c'est que les meilleures solutions pour un problème multiobjectif est un ensemble de solutions Pareto non pas une seule solution comme dans le cas de problème mono-objectif. Il faut noter que, plus d'avoir des solutions proche du front Pareto (convergence), il faut avoir aussi une bonne diversité des solutions tout au long du front. Ces deux propriétés doivent être garanties essentiellement par la méthode d'estimation. Chaque algorithme EDA utilise son propre méthode d'estimation. En explorant les méthodes d'estimation dans le domaine de statistique en peut trouver plusieurs méthodes. La méthode d'estimation doit être facile à implémenter et doit avoir une capacité pour capturer la dépendance entre les variables de problèmes et de les modéliser. Ces capacités existent dans la théorie des Copules.

En statistique, les copules peuvent modéliser les dépendances entre les variables aléatoires même si les variables sont de grande dimension [101]. L'utilisation des copules comme un estimateur de distribution dans les EDAs à fait objet de quelque travaux de recherche, Dans notre thèse on va utiliser les Copules comme un estimateur de distribution de notre algorithme CEDA pour la résolution des problèmes multiobjectifs. Dans cette thèse, premièrement on a réalisé un algorithme à estimation de distribution pour l'optimisation multiobjectif, cet algorithme repose sur les concepts des relations entre les variables de décisions d'un problème multiobjectif. Après avoir capturé les relations de dépendances entre ses variables, on modélise ces relations à l'aide d'une copule bivariée de type Archimedean. L'algorithme proposé ensuite utilise ce modèle pour créer des nouvelles solutions potentielles (individus) ces individus vont être évalués et sélectionnés à l'aide d'un algorithme connu dans le domaine de l'optimisation multiobjectif.

Pour la première contribution on a utilisé les algorithmes NSGA-II et SPEA2 et pour la deuxième où nous avons fait l'optimisation des problèmes many-objectives on a utilisé l'algorithme MOEA/D avec deux différentes méthodes de décomposition Tchybeycheff et Boundry Intersection.

La deuxième principale contribution est dans le domaine d'aide à la décision où le problème réside dans le cas si un décideur n'est pas satisfait par les solutions obtenues dans la phase d'optimisation, notre proposition utilise les modèles déjà créés et génère des nouvelles solutions sans avoir ré-exécuter la phase d'optimisation. Pour les problèmes multiobjecitfs on a juste utilisé les copules créées dans la phase d'optimisation alors que dans l'optimisation des problèmes Many-objecitves on a utilisé un algorithme hybride avec une méthode connue dans l'apprentissage automatique qui est le SVM spécialement le SVM Mono-classe. L'utilisation des SVMs nous a permis de presque supprimer l'évaluation des fonctions objectifs dans la phase de mise à jour des solutions. Cette aidée est originale et a comme avantage de surpasser une des difficultés connue dans l'optimisation Many-objective qui est le nombre d'évaluations des fonctions objectifs.

Les résultats obtenus, pour tous les benchmarks utilisés dans le point de vue de métrique de qualité utilisé ou spécialement de point de vue de propriété d'avoir des nouvelles solutions alternatives pour un problème donné soit multiobjectif ou Many-objective, montrent que notre proposition est bien adaptée pour résoudre le problème de non-satisfaction de décideur dans un problème avec même une grande dimension.

Les principales contributions dans cette thèse sont les suivantes :

— Nous élaborons un EDA basé copule pour augmenter la taille, offrir une grande diversité, et de parvenir à une convergence rapide des solutions Pareto optimales pour un problème d'optimisation multiobjectifs et Many-objectives. Nous y parvenons en exploitant les propriétés statistiques de Copules pour produire de nouvelles solutions de celles déjà existantes à travers l'estimation de leur distribution.

— Nous définissons une nouvelle métrique de performance appelée l'efficacité de génération de solution à mesurer la vitesse de générer de nouvelles solutions Pareto optimales en termes de nombre d'évaluations de la fonction objectif.

— Nous testons soigneusement CEDA sur un ensemble de problèmes références traditionnellement utilisés par la communauté, à savoir UF,CF [152] utilisant SPEA2 [157] et NSGA-II [36], et les problèmes DTLZ [35] utilisant l'algorithme MOEA/D [147].

Le reste de la thèse est organisé comme suit : Dans le chapitre 1 une définition de problèmes d'optimisation multiobjectif est donnée. Dans le chapitre 2, nous donnons un aperçu du travail effectué dans le domaine de l'optimisation multiobjectif. Dans le cha-

pitre 4, nous présentons un aperçu sur les EDAs et décrire comment ils fonctionnent généralement. Dans le chapitre 5, nous fournissons une définition mathématique de copule et certaines de leurs fonctionnalités utilisées dans EDA. Nous présentons notre base contribution CEDA dans le chapitre 6 et d'évaluer ses performances sur divers problèmes références dans le chapitre 6.

Chapitre 1

Optimisation Multiobjectif

Introduction

Plusieurs problèmes dans l'ingénierie, de l'industrie, et dans de nombreux autres domaines, impliquent l'optimisation simultanée de plusieurs objectifs. Dans la plupart des cas, les objectifs sont définis en unités incomparables, et ils présentent un certain degré de conflit entre eux (par exemple, un objectif ne peut pas être amélioré sans la dégradation d'au moins tout autre objectif). Ces problèmes sont appelés multiobjectif ou multicritères. Prenons, par exemple, une compagnie de transport qui est intéressée à minimiser la durée totale de ses routes pour améliorer le service à la clientèle. En outre, la compagnie veut aussi réduire le nombre de camions utilisés afin de réduire les coûts d'exploitation. Il est clair que ces objectifs sont en conflit parce que l'ajout de plus de camions réduit la durée des routes, mais augmente les coûts d'exploitation. En outre, les objectifs de ce problème sont exprimés en différentes unités de mesure.

Dans l'optimisation mono-objectif, il est possible de déterminer entre toute paire donnée de solutions si l'un est meilleur que l'autre. De ce fait, on obtient généralement une solution unique (à savoir, l'optimum global). Cependant, dans l'optimisation multiobjectif il n'existe pas une méthode simple pour déterminer si une solution est meilleure que les autres. La méthode la plus adoptée dans l'optimisation multiobjectif pour comparer les solutions est appelée relation de dominance de Pareto [36] qui, au lieu d'une seule solution, conduit à un ensemble de solutions avec des différents compromis entre les objectifs. Ces solutions sont appelées solutions optimales au sens de Pareto ou des solutions non-dominées.

Dans la suite, nous présentons quelques concepts généraux et notations utilisées dans

ce document. Ces concepts sont utilisés de façon répétée dans plusieurs autres chapitres, tandis que les concepts utilisés dans les chapitres particuliers, sont définis quand ils sont utilisés dans le but de faciliter la lecture de cette thèse.

1.1 Définitions

Dans cette section nous allons expliquer les différentes notations et concepts liés à l'optimisation multiobjectif, tels que la relation de dominance de Pareto, l'optimalité selon Pareto, le front Pareto et l'ensemble Pareto optimal.

1.1.1 Problème d'optimisation multiobjectif

Définition 1 *On dit qu'un problème est multiobjectif s'il comporte plusieurs objectifs qui doivent être optimisés simultanément [37]. Mathématiquement, un problème multiobjectif peut être formalisé comme suit :*

$$\begin{aligned} &\min \mathbf{F}(\mathbf{x}) && \text{où } \mathbf{F} = (f_0, f_1, \ldots, f_m) \\ &\text{tel que } \mathbf{G}(\mathbf{x}) \leq 0 \end{aligned} \tag{1.1}$$

Avec $\mathbf{x} \in \mathbb{R}^n, \mathbf{F}(\mathbf{x}) \in \mathbb{R}^m, \mathbf{G}(\mathbf{x}) \in \mathbb{R}^p$*, on a* m *fonctions à optimiser (*$m \geq 2$*) et* p *contraintes à satisfaire.* $\mathbf{x}$ *appartient à un ensemble des solutions réalisables, et à chaque solution* $\mathbf{x}$ *dans l'espace de décision est associé un vecteur objectif* $\mathbf{z}$ *dans l'espace des objectifs tel que* $\mathbf{z} = \mathbf{F}(\mathbf{x}) = (f_0(\mathbf{x}), f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))$ *(Voir Figure 1.1).*

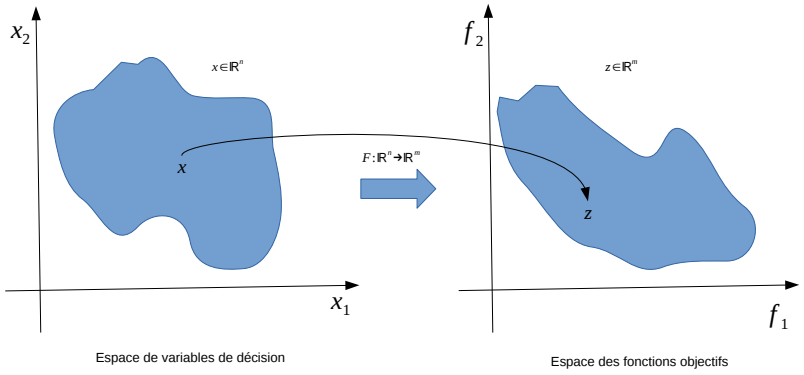


FIGURE 1.1: Espace de décision et espace objectif d'un problème d'optimisation multiobjectif.

La Figure 1.1 présente un exemple avec deux variables de décision et deux fonctions ob-

jectifs. Dans cette figure, on peut voir la façon dont la fonction $\mathbf{F}$ *projette les solutions de l'ensemble réalisable,* $\mathbf{X}$*, dans l'ensemble réalisable,* $\mathbf{Z}$*, de l'espace de la fonction objectif.*

1.1.2 Relation de dominance de Pareto et optimalité

Le but principal de l'optimisation est de trouver une solution optimale pour le problème décrit dans (1.1). Il faut noter que le problème comporte plusieurs objectifs à atteindre qui sont dans la plupart du temps contradictoires. Alors, pour comparer une solution par rapport a une autre il faut définir une nouvelle relation de comparaison. Typiquement, la relation de dominance de Pareto garantit cet but.

Définition 2 *Si on considère un problème de minimisation, un vecteur de décision* $\mathbf{u}$ ***domine faiblement*** $\mathbf{v}$ *noté (*$\mathbf{u} \preceq \mathbf{v}$*), si seulement si* $f_i(\mathbf{u}) \leq f_i(\mathbf{v}), \forall i \in \{0, 1, \ldots, m\}$ *et* $\exists j \in \{0, 1, \ldots, m\}, f_j(\mathbf{u}) < f_j(\mathbf{v})$.

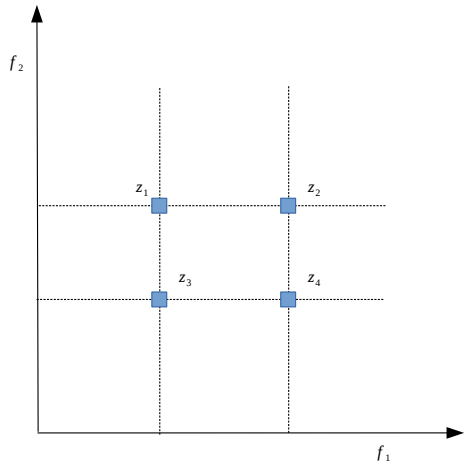


FIGURE 1.2: Description de la relation de dominance de Pareto.

La Figure 1.2 présente la relation de dominance de Pareto avec un exemple de quatre vecteurs en 2-dimensions. Le vecteur $\mathbf{z_3}$ *est strictement inférieur à* $\mathbf{z_2}$ *dans les deux objectifs, donc* $\mathbf{z_3} \prec \mathbf{z_2}$*. Le vecteur* $\mathbf{z_3}$ *également domine* $\mathbf{z_1}$ *même si par rapport à* f_1 *ces vecteurs sont égaux, mais dans* f_2*,* $\mathbf{z_3}$ *est strictement inférieur à* $\mathbf{z_1}$*. Il faut noter que certains éléments peuvent être incomparable et c'est le cas avec* $\mathbf{z_1}$ *et* $\mathbf{z_4}$*, c.-à-.d* $\mathbf{z_1} \not\prec \mathbf{z_4}$ *et* $\mathbf{z_4} \not\prec \mathbf{z_1}$*. Les comparaisons restantes sont les suivantes :* $\mathbf{z_3} \prec \mathbf{z_4}$*,* $\mathbf{z_1} \prec \mathbf{z_2}$*, et* $\mathbf{z_4} \prec \mathbf{z_2}$.

Définition 3 *Si on considère un problème de minimisation, un vecteur de décision* $\mathbf{u}$ ***domine strictement*** $\mathbf{v}$ *(*$\mathbf{u} \prec \mathbf{v}$*) si et seulement si* $f_i(\mathbf{u}) < f_i(\mathbf{v}), \forall i \in \{0, 1, \ldots, m\}$.

Définition 4 *Une solution* $\mathbf{x}^*$ *est une solution* ***Pareto optimal*** *si et seulement si il n'y a pas une autre solution admissible* $\mathbf{x}$ *où* $f(\mathbf{x})$ *domine* $f(\mathbf{x}^*)$*. De ce cas, la solution d'un problème multiobjectif est un ensemble de solutions qui sont pas dominées par aucune autre solution. On appel cet ensemble les* ***Solutions Pareto (Pareto Solutions, en anglais) PS****. L'image de cet ensemble dans l'espace objectif forme le* ***Front Pareto (Pareto Front, en anglais) PF****.*

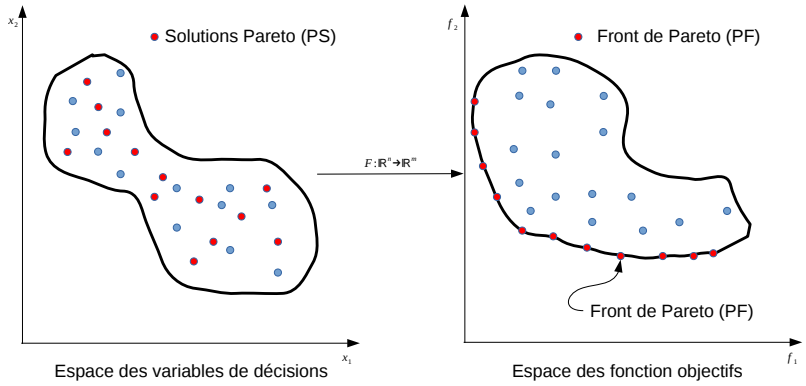


FIGURE 1.3: L'ensemble Pareto et son image dans l'espace objectif.

La Figure 1.3 explique le concept de l'ensemble Pareto optimal et son image dans l'espace objectif (le front de Pareto). Les points rouges dénotent les vecteurs Pareto optimaux. Dans l'espace de variable, ces vecteurs sont considérés comme des vecteurs de décision Pareto optimaux, tandis que dans l'espace objectif, ils sont appelés vecteurs des objectifs Pareto optimaux. Comme on peut le voir dans la figure, le front de Pareto est composé seulement par des vecteurs non dominés.

1.2 Une vue de méthodes de résolutions

Dans la plupart des problèmes d'optimisations multiobjectifs (Multiobjective optimization problem, MOP), l'ensemble Pareto optimal est composé d'un grand nombre ou même infini de solutions. Néanmoins, dans la pratique, une seule solution devrait être choisie parmi l'ensemble Pareto optimal. La personne qui a la tâche de choisir la solution la plus pratique de l'ensemble Pareto optimal est appelé le décideur (Decision Maker, DM). Les préférences du décideur sont incorporées afin d'induire un ordre total entre les solutions de Pareto. Il y a plusieurs approches dans lesquelles les préférences de DM peuvent être incorporées. Par exemple, le DM peut classer l'ensemble des objectifs en fonction de leurs importances. Une autre possibilité est d'obtenir un échantillon

du front de Pareto, puis, sélectionner une solution de cet échantillon. Une classification commune des techniques de résolution des MOPs prend en compte le moment où le DM est tenu de fournir des informations de préférence. La classification est la suivante :

Méthodes a priori Dans ce type de méthode où le décideur coopère avant le lancement du processus de résolution, le décideur doit avoir une bonne connaissance du problème traité et il doit fournir des préférences sur le problème à résoudre. Les informations données a priori aident la méthode de résolution dans sa recherche de solutions. Un exemple des méthodes a priori sont les méthodes d'agrégation qui transforment un problème multiobjectif en un problème mono-objectif. La transformation peut se faire à l'aide d'un ensemble de poids sur les objectifs, ensuite on exécute une méthode classique qui fournira la solution recherchée en une seule exécution. Il faut noter que, le choix des poids doit se faire attentivement. Ainsi, la modélisation des préférences du décideur est une tache difficile et doit être prise en compte durant l'étape de résolution. Par ailleurs, dans la plupart des cas, un décideur peut ne pas être satisfait par les solutions obtenues après la résolution, ce qui l'oblige à relancer le processus par le biais d'une autre formulation de ses préférences.

Méthodes a posteriori Dans les méthodes a posteriori on cherche à trouver les solutions Pareto optimales sans que le décideur n'intervient dans le processus de résolution (il pourrait intervenir après la résolution). Par conséquent, la méthode doit fournir au décideur la totalité de front Pareto trouvé, ensuite le décideur choisit les solutions qui lui semblent les plus appropriées. Ces méthodes ne nécessitent pas une connaissance préalable du problème, donc il n'est pas nécessaire de modéliser les préférences du décideur. Par ailleurs, le nombre de solutions obtenues peut rendre l'ensemble Pareto optimal difficile à analyser pour le décideur de plus ces méthodes soufrent du même inconvénient que les méthodes a priori si le décideur n'est pas satisfait des solutions obtenues.

Méthodes interactives Les méthodes interactives visent à impliquer le décideur tout au long du processus de recherche. Par conséquent, dans tout le processus de résolution le décideur coopère avec la méthode de résolution, le décideur peut définir ses préférences de façon claire et compréhensible. Le processus est itéré plusieurs fois jusqu'à la satisfaction du décideur. Par ailleurs, l'intervention directe du décideur dans le processus de résolution exige un long processus de recherche.

Comme la taille de l'ensemble Pareto optimal peut être infinie, dans la pratique, l'ob-

jectif d'une approche a posteriori est de trouver un ensemble d'approximation du front Pareto optimal avec une meilleure qualité.

Dans les méthodes d'optimisation interactives, il est utile de connaître les limites inférieure et supérieure du front de Pareto. Le point idéal représente la limite inférieure et il est défini par $z_i^* = min_{\mathbf{z}\in Z}(\mathbf{z_i}), \forall i = 1, \cdots, m$, Z représente l'espace des objectifs. À son tour, la limite supérieure est défini par le point nadir, qui est donnée par ${\mathbf{z_i}}^{nad} = max_{\mathbf{z}\in PF_{opt}}(\mathbf{z_i}), \forall i = 1, \cdots, m$. Afin d'éviter certains problèmes lorsque les points idéaux et nadir sont égaux ou très proches, un point est strictement meilleur que le point idéal est généralement défini. Ce point est appelé le point utopique qui est défini par ${\mathbf{z_i}}^{**} = {\mathbf{z_i}}^* - \epsilon, \forall i = 1, \cdots, m$, où $\epsilon > 0$ est un petit scalaire 1.4.

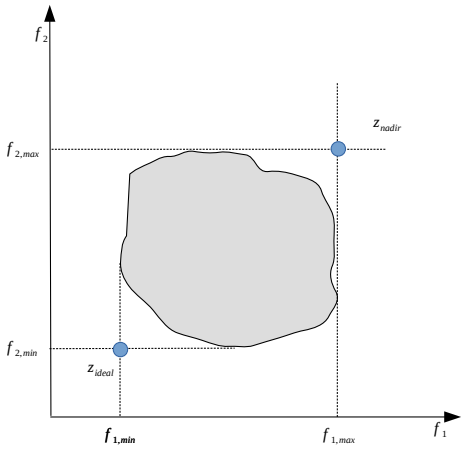


FIGURE 1.4: Point idéal et point nadir d'un problème bi-objectif.

Dans la suite de ce chapitre, nous présentons deux groupes de méthodes d'optimisation pour résoudre les problèmes d'optimisation multiobjectifs (MOPs). Le premier groupe est connu sous le nom de méthodes de programmation mathématique qui ont été proposées par la communauté de prise de décision multi-critères (Multi-Criteria Decision Making, MCDM). L'autre groupe de méthodes sont les algorithmes évolutionnaires multiobjectif (Multiobjective Evolutionary Algorithms, MOEAs), qui sont des méthodes stochastiques, proposées dans le domaine des algorithmes évolutionnaires (Evolutionary Algorithm, EA).

1.3 Programmation mathématique

Dans cette section, nous présentons une brève description de quelques techniques les plus populaires de MCDM. Ces méthodes sont organisées dans l'une des méthodes a

priori, a posteriori et des méthodes interactives.

1.3.1 Méthodes a priori

1.3.1.1 Programmation Par but

Dans cette méthode, développée par Charnes et Cooper [27], le décideur doit fixer un but qu'il souhaite atteindre pour chaque objectif. Ces valeurs sont incorporées dans le problème comme étant des contraintes supplémentaires. Ensuite la fonction objectif essaye de minimiser les écarts des cibles aux objectifs.

En outre, la méthode de programmation par but offre une flexibilité pour les cas qui ont plusieurs conflits de buts. Il faut noter que cette technique donne une solution non-dominée si le point but est choisi dans l'espace réalisable.

1.3.1.2 Méthode Lexicographique

Dans cette méthode, les objectifs sont classés par ordre d'importance par le décideur [124] (du meilleur au pire). La valeur optimale $f_i^*(i = 1, ..., m)$ est ensuite obtenue en minimisant les fonctions objectifs séquentiellement, en commençant par le plus important, puis procéder selon un ordre décroissant d'importance des objectifs. La valeur optimale trouvée pour chaque objectif est ajouté comme une contrainte pour les optimisations ultérieures. De cette façon, la valeur optimale des objectifs les plus importants est préservée. Le reste des objectifs sont considérés juste dans le cas d'obtention de plusieurs solutions optimales dans l'optimisation de l'objectif actuel, dans le pire des cas, nous devons mener m optimisations mono-objectif.

1.3.2 Méthodes a posteriori

1.3.2.1 Somme pondérée

Dans cette méthode, l'idée générale est de faire associer un coefficient de poids pour chaque fonction objective et ensuite minimiser la somme pondérée de ces objectifs [76]. De cette façon, le problème multiobjectif est transformé en un problème mono-objectif. Dans cette méthode, il est possible d'arriver à une solution optimale au sens de Pareto si les coefficients de poids sont positifs pour tous les objectifs, ou si le problème a une solution unique (voir équation 1.2). La méthode de la somme linéaire des poids peut générer des solutions Pareto optimales quelconque pour un problème d'optimisation

multiobjectif convexe (c.-à-d. si toutes les fonctions objectifs et l'espace réalisable sont convexes).

$$\min \mathbf{F} = \sum_{i=0}^{m} w_i f_i(\mathbf{x}) \quad w_i \in [0,1] \tag{1.2}$$

1.3.2.2 La méthode ϵ-Contrainte

La méthode ϵ-contrainte est une des techniques de scalarisation pour résoudre les problèmes multiobjectif. Dans cette approche, l'un des objectifs est minimisé, tandis que les autres sont utilisés comme des contraintes liées par certains niveaux admissibles ϵ_i [84]. Afin de trouver plusieurs solutions optimales de Pareto, nous devons résoudre le problème ϵ-Contrainte en utilisant plusieurs valeurs de ϵ_i. Cette méthode est un processus d'optimisation itératif dans lequel l'utilisateur doit fournir l'intervalle de l'objectif référence. Il doit être fourni l'incrément pour les contraintes imposées par ϵ. Cet incrément détermine le nombre de solutions Pareto optimales générées.

$$\begin{cases} \min f_p & f_p \text{ est la fonction préférence} \\ \text{avec } f_i \leq \epsilon_i & \text{pour } i = 0, \cdots, p-1, p+1, \cdots, m \end{cases} \tag{1.3}$$

1.3.3 Méthodes interactives

1.3.3.1 Méthode de Tchebychev

La méthode de Tchebychev proposée dans [20], est une méthode interactive basée sur la minimisation d'une valeur de fonction. Le métrique à utiliser pour mesurer les distances à un vecteur utopique d'objectif est le métrique de Tchebychev pondérée. Ainsi, le problème d'optimisation multiobjectif est transformé en un problème d'optimisation mono-objectif. Chaque solution optimale au sens de Pareto d'un problème d'optimisation multiobjectif peut être trouvée en résolvant le nouveau problème mono-objectif. Cependant, avec cette approche, certaines des solutions peuvent être des solutions Pareto optimales faibles. Cet inconvénient est résolu par la formulation du problème de minimisation de la distance comme un problème lexicographique. Dans chaque itération, la méthode Tchebychev fournit différents sous-ensembles de solutions non-dominées. Ces solutions sont constituées de $P(\approx n)$ des points représentatifs, générées en utilisant le problème Tchebychev lexicographique, à partir du quel le DM est nécessaire pour sélectionner une la plus préférée pour lui.

1.3.3.2 Méthodes de point de référence

L'approche du point de référence, proposée par Wierzbicki [138, 139], est une technique d'optimisation multiobjectif interactive basée sur la définition d'une fonction de scalarisation à atteindre. L'idée de base de cette technique est la suivante. Premièrement, le DM est invité à donner un point de référence. Ensuite, les solutions qui satisfont le mieux les niveaux d'aspiration sont calculées en utilisant la fonction de scalarization. Si le DM est satisfait de la solution actuelle, le processus interactif se termine. Sinon, le DM doit fournir un autre point de référence.

1.4 Algorithmes évolutionnaires multiobjectifs

Les problèmes d'optimisations multiobjectif peuvent être classés dans des différents groupes en fonction des caractéristiques de leurs composants, à savoir, si les fonctions objectifs sont convexes ou non, ou si l'espace de décision est continu ou discret. La liste suivante présente un classement possible pour les MOPs (Multiobjective Optimization Problem) en fonction de la nature de certains éléments clés. La classification suivante fournit une idée sur les composants communs qui guident la conception d'une méthode d'optimisation.

- Les contraintes sur les variables. Un problème d'optimisation peut ne pas avoir des contraintes sur les variables de décision, ou il peut avoir des contraintes d'égalité ou d'inégalité .
- Nature de l'espace de décision. Un problème est appelé un problème d'optimisation multiobjectif entière si certaines des variables ne peuvent prendre que des valeurs entières. D'autre part, si toutes les variables peuvent prendre des valeurs réelles, le problème est connu comme un problème d'optimisation multiobjectif continu .
- Géométrie des fonctions objectifs et espace réalisable. Un MOP est convexe si toutes les fonctions objectifs et l'espace des variables réalisables sont convexes.
- Nature des fonctions objectifs et des contraintes : lorsque toutes les fonctions objectifs et contraintes sont linéaires, le MOP est appelé linéaire. Si au moins une contrainte ou objectif est non linéaire le problème est appelé MOP non linéaire.

Dans de nombreux cas, le développement d'une nouvelle technique d'optimisation est le résultat de la nécessité de résoudre un certain genre de MOP réel. Par conséquent, la conception de cette nouvelle technique est orientée pour profiter des caractéristiques particulières du problème donné.

Dans le monde réel, il y a certains types de MOPs dans lequel les techniques de programmation mathématiques traditionnelles présentent une mauvaise performance, ou ils ne sont même pas applicables. Certains exemples sont les suivants : problèmes dans lequel les fonctions objectifs ne sont pas données dans une forme fermée, mais elles sont définies par un modèle de simulation. De plus, dans certains problèmes le hardware à optimiser est directement utilisée pour évaluer une solution. Une caractéristique commune supplémentaire dans les problèmes de la vie réelle est la présence de bruit dans les fonctions objectifs, ou des fonctions objectifs dynamiques.

Ces complexités font appel à des approches alternatives pour faire face à ces types de MOPs. Parmi ces approches, nous pouvons trouver les algorithmes évolutionnaires (EA, Evolutioanry Algorithm). Les EAs sont des méthodes de recherche stochastiques et d'optimisation qui simulent le processus de l'évolution naturelle. À la fin des années 1960, Rosenberg a proposé l'utilisation d'algorithmes génétiques pour résoudre les MOPs. Cependant, ce n'est qu'en 1984 que David Schaffer [120] a proposé la première mise en œuvre effective de ce qu'il est maintenant appelé MOEA (Multi-Objective Evolutionary Algorithm). À partir de ce moment, de nombreux chercheurs ont développé leur propre MOEAs.

Comme d'autres stratégies de recherche stochastique (par exemple, le recuit simulé, l'optimisation de colonie de fourmis, ou optimisation par essaim de particules), les EAs ne garantissent pas de trouver l'ensemble Pareto optimal mais essayent de trouver un ensemble non-dominé dont les vecteurs sont aussi proches que possible du front Pareto optimal. D'autre part, les EAs sont particulièrement bien adaptés pour résoudre les MOPs parce qu'ils travaillent en parallèle avec un ensemble de solutions possibles. Avec cette caractéristique, il est possible de trouver et avec une seule exécution plusieurs solutions de l'ensemble optimal de Pareto (ou une bonne approximation). En outre, les EAs sont moins sensibles à la forme ou à la continuité du front de Pareto, au contraire des techniques traditionnelles de programmation mathématique.

Certains chercheurs ont déterminé certaines limites d'algorithmes de programmation mathématiques traditionnelles pour résoudre MOPs. Certaines de ces limites sont les suivantes :

1. Il faut exécuter ces algorithmes plusieurs fois pour trouver plusieurs éléments de l'ensemble Pareto optimal.
2. Beaucoup de MOPS ont besoin de connaissances du domaine sur le problème à résoudre.
3. Certains de ces algorithmes sont sensibles à la forme ou la continuité du front

Pareto.

Les algorithmes évolutionnaires pour l'optimisation mono-objectif et les MOEAs partagent une structure similaire. La différence majeure est le mécanisme d'attribution de fitness puisque MOEA travaille avec des vecteurs de fitness de dimension m ($m > 2$). Comme il est souligné par des différents travaux, trouver une approximation de front Pareto est en soi un problème bi-objectif dont les objectifs sont :

— Réduire la distance entre les vecteurs générés et le front Pareto optimal.

— Maximiser la diversité de l'approximation du front Pareto trouvé.

Par conséquent, l'attribution de fitness doit tenir compte de ces deux objectifs. L'algorithme 1 décrit la structure de base d'un MOEA.

Algorithme 1 Pseudocode d'un MOEA

```
1:  t ← 0
2:  Générer une population initial P(t)
3:  tant que le critère n'est pas satisfait faire
4:      Évaluer le vecteur objectif F pour chaque individu de P(t)
5:      Attribuer un fitness pour chaque individu de P(t)
6:      Choisir parmi P(t) un groupe de parents P'(t) préférant les meilleurs
7:      Recombiner les individus de P'(t) pour obtenir les descendants P"(t)
8:      Muter des individus en P(t)
9:      Combiner P'(t) et P"(t) et sélectionner les meilleurs individus pour avoir P(t + 1)
10:     t ← t + 1
11: fin tant que
```

Généralement, la population initiale est générée d'une manière aléatoire. Cependant, si nous avons des connaissances sur les caractéristiques d'une bonne solution, il est préférable d'utiliser ces informations pour créer la population initiale. L'attribution de fitness exige le classement des individus selon une relation de préférence, puis, en attribuant une valeur de fitness scalaire à chaque individu en utilisant ce rang. La sélection pour la reproduction (ligne 6) est réalisée comme dans le cas mono-objectif, par exemple, en utilisant une sélection tournoi. En revanche, la sélection dans la ligne 9, destinée à mettre à jour les meilleures solutions (c.-à-d élitisme), utilise une relation de préférence pour enlever des solutions et maintenir la taille de la population constante.

1.4.1 Les éléments clés d'un MOEA

Dans cette section, nous allons décrire en détail les éléments qui doivent être pris en compte pour la conception d'un MOEA.

1.4.1.1 Attribution de Fitness

Dans un MOEA nous avons besoin d'un processus supplémentaire pour transformer un vecteur de fitness en une valeur scalaire. Principalement, il existe trois stratégies pour accomplir ce processus : à base de critère, à base d'agrégation et à base de préférence.

À base de critère Cette approche choisit alternativement chacune des fonctions objectifs lors de l'étape de sélection. Autrement dit, pour sélectionner un individu ou groupe d'individus seul un objectif est considéré. Par exemple, l'algorithme Vector Evaluated Genetic Algorithm (VEGA) [120] divise la population en m sous-populations de même taille, et un objectif différent est utilisé pour attribuer le fitness de chaque sous-population.

À base d'agrégation Dans cette méthode, les fonctions objectifs sont agrégées ou combinées en une valeur scalaire unique. Pendant le processus d'optimisation, les paramètres sont modifiés systématiquement pour générer des différents éléments de l'ensemble Pareto optimal. Il faut noter que, même si une approche à base d'agrégation peut être formulée comme une relation de préférence, les solutions ne sont pas comparées dans l'espace objectif. En d'autres termes, les vecteurs sont mappés à partir de $\mathbb{R}^m$ vers $\mathbb{R}$ avant la comparaison.

À base de préférence Dans ce schéma, une relation de préférence est utilisée pour induire un ordre partiel de la population dans l'espace objectif. Ensuite, un score scalaire (rang) est attribué à chaque solution basée sur la façon avec laquelle la solution se compare par rapport aux autres solutions. Par exemple, les méthodes basées sur le rang de dominance comptent le nombre d'individus par lequel un individu donné est dominé. Dans les méthodes basées sur le nombre de dominances, le fitness d'un individu correspond au nombre d'individus qu'il domine. La dominance de Pareto est la relation de préférence la plus adoptée en MOEAs.

1.4.1.2 Élitisme

L'élitisme est le mécanisme destiné à prévenir la perte des meilleures solutions trouvées lors de la recherche en raison d'effets stochastiques. Ce concept joue un rôle majeur dans les MOEAs. Dans l'optimisation multiobjectif, la mise en œuvre de l'élitisme est plus complexe que dans optimisation mono-objectif. En conséquence des ressources mémoire limitées, si des solutions non-dominées de plus surviennent que ceux qui peuvent être stockées, alors quelles sont les solutions qui doivent être rejetées ? Par conséquent, la stratégie élitiste adoptée détermine si le fitness est globalement convergent ou non. Actuellement, nous pouvons distinguer principalement deux approches pour mettre en œuvre l'élitisme. L'une d'elles est de combiner l'ancienne et la nouvelle population, et ensuite utiliser une sélection qui préserve les meilleures solutions dans la prochaine génération [36]. L'autre approche est de maintenir un ensemble externe d'individus appelés archives qui stockent les solutions non-dominées trouvées au cours du processus de recherche. La figure 1.5 illustre ces deux approches.

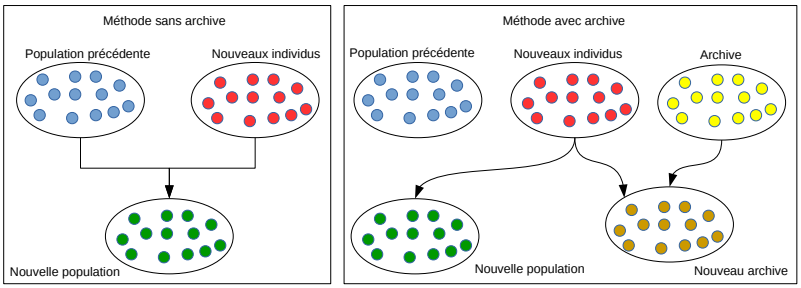


FIGURE 1.5: Description des méthodes sans archive et méthodes avec archive.

1.4.1.3 Estimateurs de densité

Un des buts en MOEAs est d'obtenir un ensemble de solutions non-dominées qui sont bien distribuées le long du front de Pareto. Dans ce qui suit, nous décrivons quelques techniques pour maintenir la diversité dans la population.

Partage de fitness L'objectif de partage du fitness est de former et de maintenir des sous-populations (niches) réparties sur l'espace objectif. L'idée est de considérer le fitness comme une ressource qui doit être partagée entre les individus dans la même niche. Ainsi, plus il y a d'individus dans la niche, moins est le fitness attribué à chaque individu.

Formellement, le fitness partagé f_{s_i} de l'individu i est défini par :

$$f_{s_i} = \frac{f_i}{\sum_{j=1}^{N} \phi(d_{ij})'} \tag{1.4}$$

où f_i est le fitness de l'individu i et $\phi(d_{ij})$ est la fonction de partage, définie par :

$$\phi(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{sh}}\right) & , \quad d_{ij} < \sigma_{share} \\ 0 & , \quad sinon \end{cases} \tag{1.5}$$

où σ_{share} est le rayon de la niche et d_{ij} est la distance entre les individus i et j.

Hypergrilles Un hypergrille divise l'espace objectif dans des régions appelées hypercubes. Chaque solution non-dominée occupe un hypercube comme il est montré dans la figure 1.6. L'idée est seulement d'accepter des solutions non-dominées appartenant à un hypercube peu peuplé. Bien que le nombre de divisions dans le hypergrille dans chaque dimension sont constant, la position et l'extension de la grille peuvent être adaptées au cours du processus de recherche.

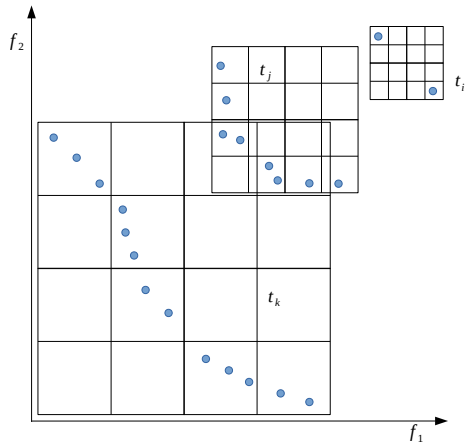


FIGURE 1.6: L'hypergrille pour maintenir la diversité dans l'archive.

Clustering L'objectif d'un algorithme de clustering est de partitionner un ensemble de points de telle sorte que : (1) chaque groupe contient des points très proches les uns des autres ; et (2) les points d'un groupe sont très différents des points des autres groupes. Dans un MOEA, nous utilisons le clustering pour préserver la diversité dans l'archive et réduire sa taille. Ce processus est composé de trois étapes (figure 1.7) :

1. Partitionner l'archive en utilisant un algorithme de clustering.
2. Sélectionnez un individu représentatif de chaque groupe.
3. Enlever tous les autres individus dans le cluster.

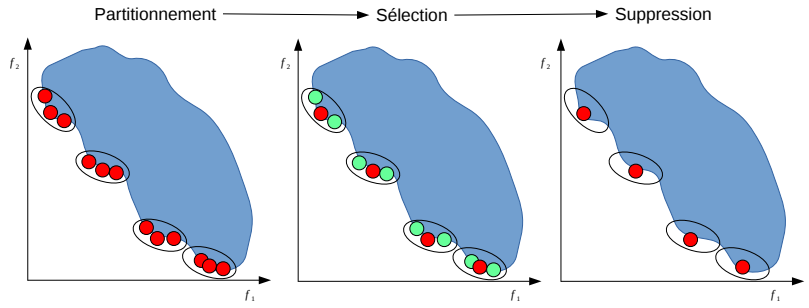


FIGURE 1.7: Les étapes des méthodes de clustering dans le processus d'optimisation.

1.4.2 MOEAs basés sur l'optimalité de Pareto

Multiobjective Genetic Algorithm (MOGA) : Fonseca et Fleming [100] ont proposé Multiobjective Genetic Algorithm (MOGA), qui est basé sur le schéma proposé par Goldberg [55]. Cet algorithme classe la population sur la base de non-dominance. Ainsi, le rang d'un individu x^i dans la génération t est égal au nombre de solutions, $p(x^i, t)$, par lesquelles il est dominé $rang(x^i, t) = 1 + p(x^i, t)$.

Non-dominated Sorting Genetic Algorithm (NSGA) : Srinivas et Deb [130] ont implémenté l'idée de Goldberg d'une manière plus simple. Non-dominated Sorting Genetic Algorithm (NSGA) classe la population dans des différentes couches ou fronts non-dominés par rapport au non-dominance. Le premier front (le meilleur classement) est composé par les individus non-dominés de la population actuelle. Le deuxième front est l'ensemble d'individus non-dominés à l'exclusion des individus du premier rang. En général, chaque front est calculé uniquement avec les individus non classés dans la population. Deb et al. [37] plus tard, ont proposé une nouvelle version de l'algorithme appelé NSGA-II. Cet algorithme améliore l'efficacité de NSGA originale en réduisant le nombre de fois que la population a besoin d'être classée, et intègre un système de sélection élitiste, ainsi que d'un opérateur de comparaison Crowding.

Niched Pareto Genetic Algorithm (NPGA) : Horn et Nafpliotis [33] ont combiné la sélection par tournoi et la dominance de Pareto. Dans cette méthode, deux individus sont sélectionnés au hasard afin de les comparer à un sous-ensemble de la population. L'individu non-dominé est choisi en tant que parent. D'autre part, si les

deux individus sont dominés ou non-dominés, alors le gagnant est choisi à l'aide d'une fonction de partage de fitness.

Strength Pareto Evolutionary Algorithm (SPEA) : Strength Pareto Evolutionary Algorithm (SPEA) a été développé par Zitzler et Thiele [156] comme un moyen pour combiner les techniques les plus réussies des différents MOEAs. SPEA utilise les individus stockés dans les archives pour classer les individus dans la population actuelle. Pour chaque individu dans l'archive une valeur s est calculée appelée strength (force) qu'est égale au nombre d'individus de la population qui sont dominés par l'individu de l'archive correspondant. Le fitness de chaque individu x est calculé en additionnant s (strength) de tous les membres de l'archive qui dominent x. Ce schéma tente de guider la recherche vers le front de Pareto et, dans le même temps, préserve la diversité des solutions non-dominées.

Conclusion

On a présenté dans ce chapitre les définitions et les principaux concepts utilisés dans le domaine d'optimisation multiobjectif. Comme il a été définit qu'un problème multiobjectif est un problème qui a plusieurs objectifs à optimiser. L'utilisation de mot optimiser vient de dire qu'il faut choisir une solution qui minimise où maximise les objectifs de problème alors que dans le cas d'un problème qui contient un ensemble d'objectifs on ne peut pas trouver une solution qui optimise une fonction objectif au même temps on trouve qu'il optimise l'autre (dans le cas de deux objectifs) objectif. L'apparition de ce problème nous a mené à utiliser un nouveau concept de comparaison entre les solutions ce concept s'appelle la dominance de Pareto. Par conséquent, si on utilise la relation de dominance pour comparer les solutions on trouve que les meilleures solutions qu'il faut choisir sont un ensemble de solutions dit les solutions non-dominées ou solutions Pareto, l'application des fonctions objectifs sur les solutions Pareto produit ce qu'on l'appelle Front Pareto.

Chapitre 2

Algorithmes évolutionnaires pour l'optimisation multiobjectif

Introduction

Les algorithmes évolutionnaires (Evolutionary Algorithm, EA) sont des heuristiques qui utilisent la sélection naturelle comme leur moteur de recherche pour résoudre les problèmes. L'utilisation de l'EA pour les tâches de recherche et d'optimisation est devenue très populaire au cours des dernières années avec un développement constant de nouveaux algorithmes, les réalisations théoriques et de nouvelles applications. Un des domaines de recherche émergente dans laquelle EA sont devenus de plus en plus populaire est l'optimisation multiobjectif. Dans les problèmes d'optimisation multiobjectif, nous avons deux ou plusieurs fonctions objectifs à optimiser au même temps, au lieu d'avoir une seule. En conséquence, il n'y a pas de solution unique aux problèmes d'optimisation multiobjectifs, mais à la place, on trouve toutes les solutions bon compromis disponibles (la soi-disant Pareto ensemble optimal).

La première mise en œuvre d'un algorithme évolutionnaire multiobjectif (MOEA) remonte au milieu des années 1980. Depuis, une quantité considérable de recherches ont été faites dans ce domaine, maintenant connu comme Algorithme évolutionnaire pour l'optimisation multiobjectif (Evolutionary Multiobjective Optimization, EMO). L'importance croissante de ce domaine se traduit par une augmentation significative (principalement au cours des dix dernières années) de documents techniques à des conférences internationales et des revues, des livres, des séances spéciales à des conférences internationales et des groupes d'intérêt sur Internet [32].

La motivation principale de l'utilisation des EA pour résoudre des problèmes d'optimisation multiobjectifs est que les EA traitent simultanément avec un ensemble de solutions possibles (dite population) qui nous permet de trouver plusieurs membres de l'ensemble Pareto-optimale en une seule exécution de l'algorithme, au lieu d'effectuer une série d'essais séparés comme dans le cas des techniques mathématiques classiques. En outre, les EAs sont moins sensibles à la forme ou la continuité du front de Pareto (par exemple, ils peuvent facilement faire face à fronts de Pareto discontinues et concaves), alors que ces deux questions sont une réelle préoccupation pour les techniques de programmation mathématique.

Dans ce chapitre, nous allons faire une synthèse des méthodes de résolutions des problèmes multiobjectif. Dans la littérature, les méthodes utilisées pour résoudre ces problèmes se divisent en deux classes : les méthodes exactes et les méthodes qui utilisent les méta-heuristiques. La différence entre les méthodes exactes et les méthodes de métaheuristiques est que les méthodes exactes trouvent dans la plupart du temps des solutions exactes pour le problème mais consomme un temps de calcul énorme et il ne peut que trouver une solution pour un problème multiobjectif ce qui est le contraire pour les méthodes de résolutions qui utilisent les métaheuristiques essentiellement les algorithmes évolutionnaires qui trouvent un ensemble approximatif de solutions Pareto avec un temps acceptable.

2.1 Algorithmes évolutionnaires multiobjectif

Dans cette section, plusieurs algorithmes de l'état de l'art dans des différents frameworks d'optimisation multiobjectifs sont décrits.

2.1.1 Framework basé préférence

Les frameworks basés préférence sont les méthodes classiques de traitement des MOPs. L'idée de base de ce framework est d'agréger les multiples objectifs contradictoires d'un MOP en un problème d'optimisation mono-objectif ou d'utiliser les connaissances de préférence des problèmes afin que les optimiseurs peuvent se concentrer sur l'optimisation de certains objectifs.

L'approche la plus fondamentale dans ce framework est d'agréger les objectifs contradictoires dans un seul objectif par le biais d'une méthode de somme pondérée. Cette méthode combine tous les objectifs contradictoires en multipliant chaque objectif avec une

valeur de poids prédéfinie. La méthode de métrique pondérée ou de Tchebychev pondérée est une autre approche qui combine les objectifs dans un seul objectif. Dans cette dernière, l'objectif est de minimiser les métriques de distance pondérées, où la métrique distance mesure la distance d'une solution à une solution idéale. Dans [143], Haimes et al. ont proposé une méthode pour optimiser seulement l'un des objectifs et de garder les autres objectifs dans les valeurs prédéfinies par l'utilisateur. Dans cette méthode, les différentes solutions optimales peuvent être générées avec des valeurs prédéfinies par l'utilisateur.

Le principal inconvénient de ce framework est qu'il ne parvient pas à atteindre le but commun d'optimisation multiobjectif qui est d'obtenir un ensemble de compromis et de solutions diverses en un seul passage de simulation. De plus, il est nécessaire de fournir des connaissances de préférence des MOPs, telles que les valeurs de poids appropriées ou des valeurs prédéfinies par l'utilisateur, aux optimiseurs. La recherche dans ce type de framework se concentre principalement sur l'étude de la façon d'utiliser au mieux l'information de préférence dans l'optimisation.

2.1.2 Framework basé dominance

Les algorithmes dans ce framework optimisent tous les objectifs contradictoires de MOPs simultanément par l'attribution d'un fitness à chaque solution. Cette idée a été suggérée par Goldberg [55]. Notons que cette méthode n'a pas utilisé la simulation pour prouver sa pertinence dans la manipulation des MOPs. Parmi les premiers MOEAs remarquables dans ce framework, nous citons l'algorithme génétique multiobjectif (MOGA), a été proposé par Fonseca et Fleming [100]. Dans MOGA, deux étapes importantes ont été conçues pour déterminer le fitness d'une solution. En premier, le fitness d'une solution est calculée en fonction du nombre d'autres solutions quelle domine. Ce fitness est utilisé pour déterminer le rang des solutions. Deuxièmement, le fitness des solutions dans le même rang est partagé par un mécanisme de partage de fitness. L'utilisation de ces deux étapes rend MOGA capable de maintenir un ensemble de solutions non-dominées en une seule exécution de la simulation.

Srinivas et Deb [130] ont employé un framework similaire à celle de MOGA en introduisant un nouveau mécanisme de classement et de partage. L'algorithme proposé est désigné comme l'algorithme génétique de tri non-dominé (NSGA). Au lieu de compter le nombre des individus qui domine chaque solution, Srinivas et Deb ont proposé un mécanisme de classement qui classe les solutions selon le niveau de domination. Le premier niveau comprend toutes les solutions non dominées. Ensuite, les solutions marquées

comme premier niveau sont ignorées. La deuxième série de solutions non-dominées dans la population sont identifiées et marquées comme le deuxième niveau (front). Ceci continue jusqu'à ce que le classement n'a plus de solutions stockées dans la population. La diversité est maintenue grâce à une méthode de partage de fitness.

MOGA et NSGA souffrent de plusieurs limitations. Premièrement, ils sont des approches non-élitistes. Deuxièmement, il est nécessaire de spécifier un paramètre de partage. Troisièmement, l'algorithme NSGA a une complexité de calcul très élevée. Afin de développer un algorithme plus efficace pour l'optimisation multiobjectifs, certains chercheurs y ont incorporé un mécanisme d'élitisme, qui est un archive externe de solutions, dans les algorithmes d'optimisation.

Dans [154], Zitzler et Thiele ont proposé une MOEA élitiste appelée Strength Pareto Evolutionary Algorithm (SPEA). Un archive externe est créée pour maintenir un ensemble de solutions non dominées (population externe) trouvées lors des processus évolutifs. Dans chaque génération, une nouvelle élite non dominée trouvée dans la population actuelle est archivée tandis que les solutions dominées dans la population externe seront rejetées. Une fois que la population externe atteint un nombre maximum de solutions admises, les solutions surpeuplées, déterminées par un algorithme de clustering, seront rejetées. Pour l'affectation du fitness, une valeur de fitness sera affectée aux deux populations externe et actuelle. Le fitness d'une solution dans la population externe est proportionnel au nombre de solutions dans la population actuelle qui est dominée par la solution dans la population externe. En revanche, le fitness d'une solution dans la population actuelle est proportionnel à la somme de fitness des solutions dans la population externe qui domine la solution dans la population actuelle. Dans [158], Zitzler et al. ont proposé SPEA2, qui est une version améliorée de SPEA. Dans SPEA2 une affectation de fitness améliorée, l'archivage ,et un mécanisme de préservation de diversité ont été proposées.

Dans [36], Deb et al. présentent NSGA-II, qui est une version améliorée de NSGA. NSGA-II conserve également un ensemble de solutions archivées depuis le début de l'évolution. Au lieu de stocker les solutions non dominées seulement comme SPEA, NSGA-II stocke tous les parents (N solutions) et les enfants (N solutions) des solutions. Par la suite, un tri non-dominé est appliqué aux solutions d'archivage entières (2N solutions). Les solutions sont classées selon le rang de dominance. Les meilleures solutions non-dominées sont marqués comme premier rang. Les meilleurs deuxièmes solutions non-dominées sont marquées comme second rang, et ainsi de suite. Les meilleures N solutions avec les rangs inférieurs sont choisies comme solutions parents et seront sou-

mises à l'évolution de la prochaine génération. Comme il faut que N solutions soient sélectionnées comme des solutions parents, certaines des solutions à rang particulier ne convient pas d'être dans la population parent. Dans ce cas, une mesure de distance de crowding est utilisée pour déterminer les solutions qui ont moins de distances pour devenir des solutions parents. Comparé à la NSGA, NSGA-II a une complexité de calcul inférieure, utilise une approche élitiste, et n'a pas besoin de spécifier un paramètre de partage. Actuellement, NSGA-II est l'un des MOEAs les plus célèbres qui a été mis en œuvre pour résoudre de nombreux problèmes du monde réel et de servir comme un algorithme de base pour MOEAs. Cependant, la performance de l'optimisation de NSGA-II est moins bonne dans les problèmes avec plus de trois objectifs. Ceci est parce que ses mécanismes de classement et de crowding sont inefficaces pour différencier la supériorité des solutions à des problèmes à multiples objectifs (many-objective).

Le framework à base de dominance de l'optimisation multiobjectif est devenu l'un des principaux domaines de recherche de la dernière décennie. La possibilité d'obtenir un ensemble de solutions de compromis en un seul passage de simulation détermine la pertinence de cette approche pour l'optimisation multiobjectif. De plus, la diversité des solutions peut être préservée en tenant compte de la distribution des solutions dans une population maintenue. Cependant, un inconvénient majeur de cette approche est que la pression de sélection est affaiblie avec l'augmentation du nombre de fonctions objectifs. Par conséquent, cette approche ne convient que pour résoudre les problèmes avec deux ou trois fonctions objectifs. En outre, il est nécessaire de déterminer le réglage d'un paramètre de partage dans certains systèmes de préservation de diversité.

Depuis, de nombreuses variantes de MOEAs dans le framework basé sur la dominance ont été proposées. La plupart d'entre elles suivent un processus quasi-similaire que les MOEAs élitistes susmentionnées. Afin de renforcer la capacité d'optimisation des MOEAs, plusieurs aspects d'algorithmes basés sur la dominance sont devenus les principales préoccupations de recherche. Ces aspects sont MOEAs basés sur les indicateurs [21], MOEAs hybrides [44, 86], MOEAs mémétiques [80, 85], MOEAs basé sur la coévolution [53], MOEAs adaptatifs [68, 131], et MOEAs parallèles [141]. En outre, la mise en œuvre des MOEAs à base de dominance ont été largement utilisées pour faire face aux diverses MOPs, y compris les MOPs dynamiques [13, 47], MOPs avec contraintes [24, 140], MOPs à multiples objectifs [22, 127] , et MOPs multimodales [38, 145]. En plus d'employer les GA (Genetic Algorithm) comme un algorithme de recherche, beaucoup d'autres EAs ont été mises en œuvre dans le framework à base de dominance pour l'optimisation multi-objectif. Citons l'évolution différentielle (DE) [58, 112], optimisation par essaim de particules (PSO) [2, 92], EDA [42, 70], l'optimisation par colonie de fourmis

(ACO) [41, 50], le système immunitaire artificiel [57, 67], la programmation génétique (GP) [146, 151], et la stratégie d'évolution (ES) [69, 78]. Un examen plus détaillé des MOEAs basées dominance peut être consulté en [31, 137, 153].

2.1.3 Framework basé sur la décomposition

Le framework basé sur la décomposition décompose un MOP en plusieurs sous-problèmes, par la suite l'algorithme optimise tous les sous-problèmes simultanément. La dominance de Pareto est partiellement appliquée ou totalement éliminées dans ce cadre.

Dans [72], un algorithme génétique de recherche locale multi-objectifs (MOGLS) a été proposé. Dans MOGLS, les multiples objectifs d'un MOP sont agrégés en utilisant une approche de somme pondérée. Pendant le processus de sélection, une paire de solutions parents est sélectionnée pour effectuer les opérations génétiques (croisement et mutation) afin de générer un descendant. Le descendant est ensuite optimisé en utilisant la recherche locale où la fonction de fitness de descendant est une fonction agrégée de somme pondérée. Dans l'exécution de chaque recherche locale, un nouveau vecteur d'information de poids est généré aléatoirement. Dans MOGLS, une archive externe, qui stocke les solutions non dominées est également mise en œuvre. Après une recherche locale, si le descendant (fils) est non-dominé par un parent ou des solutions archivées, le descendant est ajouté à l'archive. Les solutions, dans l'archive, qui sont dominées par les solutions nouvellement ajoutées sont éliminées. Dans cet algorithme, la dominance de Pareto est partiellement appliquée.

Dans [74], une autre version de MOGLS a été proposée. Dans cette version, l'objectif était d'optimiser simultanément un MOP au lieu de trouver une solution non dominée à chaque itération comme dans [72]. Les différentes fonctions objectifs d'une MOP sont agrégées en utilisant une fonction de scalairisation de Tchebycheff pondérée. Dans chaque génération, un vecteur de pondération généré de manière aléatoire est mis en œuvre. Ensuite, un ensemble de solutions avec la meilleure fonction de scalairisation sont choisie pour former une population temporaire. Une paire de solutions parents est choisie parmi la population temporaire et des opérations génétiques sont réalisées pour générer un descendant. Par la suite, le descendant généré est amélioré localement.

Dans [104], une recherche locale en deux phases pour l'optimisation bi-objectif a été proposée. Cet algorithme tente d'optimiser une MOP par l'optimisation d'un ensemble de fonctions objectifs modifiées. Les fonctions objectifs modifiées sont construites en utilisant une fonction d'agrégation linéaire pondérée. L'idée est inspirée de l'utilisation

de recherche locale pour optimiser plusieurs fonctions d'optimisation scalaires. Après qu'une fonction scalaire est optimisée, la prochaine optimisation est effectuée pour la deuxième fonction scalaire. Le processus se poursuit jusqu'à ce que toutes les fonctions scalaires sont optimisées. Au cours du processus d'optimisation, la solution obtenue dans la fonction scalaire précédente est utilisée comme point de départ pour la résolution de fonction scalaire suivante. Cet algorithme a montré des performances prometteuses dans la résolution de TSP(Traveling Salesman Problem) bi-objectif. Cependant, la question de la diversité n'a pas été soigneusement étudiée et examinée.

Dans [147], Zhang et Li ont proposé un MOEA sur la base de décomposition (MOEA/D). Dans MOEA/D, un MOP est décomposé en plusieurs sous-problèmes dans lesquels chaque sous-problème est construit en utilisant une somme pondérée ou une approche de Tchebycheff. Ainsi, chaque sous-problème est un problème d'optimisation à objectif scalaire. Pour l'agrégation, un ensemble de vecteurs de poids uniformes est généré. Ce vecteur de poids sert de critère, en termes de distance euclidienne, pour déterminer les relations de voisinage entre les sous-problèmes. Les sous-problèmes sont alors optimisés en considérant uniquement les solutions de voisinage. Cela se fait en effectuant des opérations génétiques entre un sous-problème avec ses solutions de voisinage. La meilleure solution trouvée, en termes de la valeur de fitness agrégée, est alors mise à jour pour les sous-problème et ses solutions de voisinage. Dans MOEA/D, il n'y a pas d'ensemble pour l'archivage et l'élitisme. Même s'il n'y a pas d'élitisme évident appliqué, il est implicitement réalisé lorsque les voisins les plus proches (parents) sont mis à jour en comparant leurs valeurs de fitness avec celles des descendants (solution nouvellement générée pour un sous-problème). Avec ce mécanisme, la pression de sélection des problèmes à multiples objectifs rencontrée par les MOEAs basés sur la dominance est résolu car le fitness d'une solution dépend uniquement de la valeur objectif agrégée. En outre, il est inutile de spécifier un schéma de préservation de diversité car la diversité est préservée dans les vecteurs de pondération prédéfinis. Le principal avantage de MOEA/D est qu'un algorithme pour l'optimisation mono-objectif peut être appliqué directement à optimiser les sous-problèmes construits tandis que son framework peut préserver l'ensemble de solutions diverses.

Une version améliorée du MOEA/D a été proposée dans [89]. Tout d'abord, au lieu d'utiliser les algorithmes génétiques comme algorithme de recherche, l'évolution différentielle a été intégrée dans MOEA/D pour former un nouvel algorithme appelé MOEA/D-DE. Avec les opérateurs de DE , MOEA/D-DE est en mesure d'explorer plus efficacement l'espace de recherche en particulier lorsque l'ensemble Pareto est compliqué. Ceci parce que l'opérateur DE, est capable de générer un grand nombre de descendants car l'opé-

rateur crée un descendant en utilisant trois solutions parents. Ainsi, la capacité d'exploration de l'algorithme peut être améliorée. Deuxièmement, deux structures voisines sont proposées pour équilibrer l'exploration et l'exploitation de la recherche. L'une des structures est que le nombre maximal de sous-problèmes qui sont autorisés à être mis à jour par un descendant est équivalent à la taille du voisinage (T). Cela est identique à la structure de voisinage dans MOEA/D originale. Cette structure peut réduire la préservation de la diversité si un descendant est fortement supérieur à celle de son parent et les voisins de son parent. Afin de réduire cet effet, une seconde structure de voisinage est proposée. Cette structure limite le nombre de sous-problèmes à être mis à jour par un descendant à T_B où T_B est beaucoup plus petite que T.

2.2 Algorithme à estimation de distribution multiobjectif (MOEDA)

Les algorithmes à estimation de distribution (Estimation of Distribution Algorithm, EDA) ont été introduits comme un nouveau paradigme dans le domaine des algorithmes évolutionnaires [99]. Dans EDAs, les fils (descendants) sont produits par échantillonnage de la distribution de probabilité estimée de la population parent [82]. Récemment, plusieurs EDAs ont été proposés dans le cadre de l'optimisation multiobjectif.

Bosman et Thierens [19] ont présenté une mixture de distribution comme un modèle probabiliste où à chaque composant dans la mixture est une factorisation univariée. Cet algorithme est connu comme Multiobjective Mixture-based Iterated Density Estimation Evolutionary Algorithm (MIDEA). MIDEA a l'avantage de préserver la diversité en raison de sa capacité d'exploration largement répandue vers le front Pareto optimal. Cet algorithme peut servir d'une référence pour les algorithmes de MOEDA pour sa simplicité, sa rapidité et son efficacité. Dans une autre étude, Laumanns and Ocenasek [83] ont incorporé un algorithme d'optimisation bayésienne basé sur l'arbre de décision binaire comme une technique de construction de modèle - Bayesian Multi-objective Optimization Algorithm (BMOA) - pour approximer l'ensemble des solutions Pareto optimales.

Li et al. [88] ont proposé un EDA hybride (MOHEDA) en intégrant une recherche locale qui est basée sur la méthode de somme pondérée tandis que sa manipulation de contrainte est basée sur une méthode de réparation aléatoire pour résoudre le problème sac-à-dos 0/1. Une méthode de classification stochastique a également été introduite afin de préserver la diversité des solutions. Les résultats ont montré que MOHEDA

surpasse plusieurs algorithmes de l'état-de-l'art. Zhang et al. [149] ont proposé Regularity Model-based Multiobjective Estimation of Distribution Algorithm (RM-MEDA) en tenant compte de la régularité dans la construction du modèle probabiliste. Une analyse en composantes principales a été utilisée pour extraire les modèles de régularité des solutions candidates de recherches précédentes. Cet algorithme a montré de bonnes performances dans les benchmarks avec liaison de variable non linéaire pour un certain nombre de variables scalables. Okabe et al. [103] ont introduit les diagrammes de Voronoi pour construire le modèle de probabilité dans un algorithme appelé Voronoi-based EDA (VEDA). VEDA est en mesure d'ajuster le processus de reproduction sur la base de structure de problème et, en même temps, il prend avantage de la structure du problème en estimant la distribution de probabilité la plus appropriée. Pour plus d'éclaircissement, le chapitre suivant donne une vue plus détaillée des Algorithmes à estimation de distributions.

2.3 Algorithmes associés

2.3.1 Algorithme génétique de tri par non-dominance II

NSGA-II (Non-dominated Sorting Genetic Algorithm II) est l'une des procédures de MOEA couramment utilisées qui tente de trouver des solutions Pareto-optimale d'un problème d'optimisation multiobjectif NSGA-II a les trois caractéristiques suivantes :

1. Il utilise un principe élitiste.
2. Il utilise un mécanisme de préservation de diversité explicite.
3. Il insiste sur les solutions non-dominées.

À toute génération t, la population des descendants (par exemple, Q_t) est d'abord créée en utilisant la population parent (par exemple, P_t) et les opérateurs génétiques habituels. Par la suite, les deux populations sont combinées pour former une nouvelle population (par exemple, R_t) de taille $2N$. Ensuite, la population R_t est classée en différents fronts non-dominé. Par la suite, la nouvelle population est remplie par des solutions de différents fronts non-dominées, une à la fois. Le remplissage commence par le premier front non-dominé (de classe 1) et se poursuit avec les points de second front non-dominé, et ainsi de suite. Puisque la taille de la population globale R_t est $2N$, tous les fronts ne peuvent pas être inclus dans les N emplacements disponibles dans la nouvelle population. Tous les fronts qui ne peuvent pas être inclus sont alors supprimés. Lorsque le dernier front permis est envisagé, il peut exister plusieurs points dans le front que les

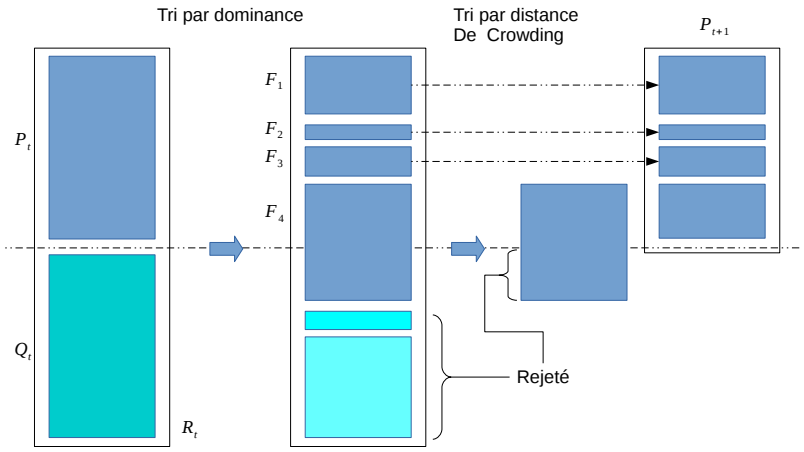


FIGURE 2.1: Représentation schématique de l'algorithme NSGA-II.

emplacements restants dans la nouvelle population. Ce scénario est illustré dans la Figure 2.1. Au lieu d'exclure arbitrairement certains membres du dernier front, les points qui réaliseront la diversité la plus élevée des points sélectionnés seront choisis [36].

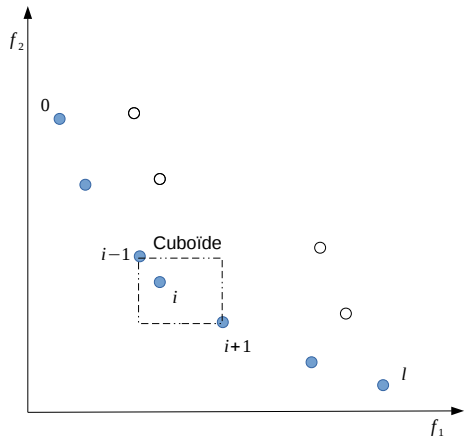


FIGURE 2.2: Calcul de distance Crowding.

Le tri de Crowding des points de dernier front qu'on a pu les accueillir entièrement est atteint dans l'ordre décroissant de leurs valeurs de distance Crowding, et les points de la partie supérieure de la liste ordonnée sont choisis. La distance de Crowding d_i du point i est une mesure de l'espace objectif autour de i qui n'est pas occupée par aucune autre solution dans la population. Ici, nous calculons tout simplement cette quantité d_i en estimant le périmètre du cuboïde (Figure 2.2) formé en utilisant les voisins les plus proches dans l'espace objectif (Nous appelons cela la distance Crowding).

Les étapes et le déroulement de l'algorithme NSGA-II peuvent être résumé dans le pseudo-code de l'algorithme 2.

Algorithme 2 Pseudocode de NSGA-II

```
t ← 0
Générer une population initiale P(t) de solutions aléatoires
tant que le critère n'est pas satisfait faire
    Sélectionner les parents pour faire les opérations génétiques
    Produire Q(t) en appliquant les opérations génétiques
    R(t) ← P(t) ∪ Q(t)
    Classer R(t) en différents fronts non-dominé
    Sélectionner les meilleurs N individus de R(t)
    Mettre les meilleurs individus dans P(t+1)
    t ← t + 1
fin tant que
```

2.3.2 Algorithme évolutionnaire de force Pareto SPEA2

La première version de l'algorithme SPEA utilise une population régulière et une archive. Les étapes suivantes sont effectuées par itération en commençant par une population initiale et une archive vide . Premièrement, tous les membres de la population non-dominés sont copiés dans l'archive ; des individus ou les redondances (concernant les valeurs des objectifs) dominées sont retirées de l'archive lors de cette opération de mise à jour. Si la taille de l'archive mise à jour dépasse une limite prédéfinie, d'autres membres de l'archive sont supprimés par une technique de clustering qui conserve les caractéristiques de front non-dominé. Ensuite, les valeurs de fitness sont affectées aux membres de l'archive et de la population :

- Chaque individu i dans l'archive est affecté une valeur Strength $S(i) \in [0, 1)$, qui en même temps représente la valeur de fitness $F(i)$. $S(i)$ est le nombre de membres de la population j qui sont dominée par, ou égaux à, i par rapport aux valeurs des objectifs, divisé par la taille de population+1.
- Le fitness $F(j)$ d'un individu j dans la population est calculé en additionnant les valeurs $S(i)$ de tous les membres de l'archive i qui dominent ou égaux à j, et en ajoutant '1' à la fin.

La prochaine étape représente la phase de sélection où les individus de l'union population et archives sont sélectionnés par le biais de tournois binaires. Noter que le fitness

doit être minimisé, c.-à-d, chaque individu dans l'archive a plus de chances d'être sélectionné que tous les membres de la population. Enfin, après le croisement et mutation l'ancienne population est remplacée par la population de descendant obtenue [157].

Algorithme 3 Pseudocode de l'algorithme SPEA2

```
1: générer une population initiale P_0
2: créer une archive vide P̄_0 = ∅
3: t←0
4: tant que critère d'arrêt non satisfait faire
5:     calculer les fitness des individus de P_t et P̄_t
6:     Copier tous les individus de P_t et P̄_t dans P̄_{t+1}.
7:     si Taille de P̄_{t+1} > N̄ alors
8:         réduire P̄_{t+1} par l'intermédiaire de l'opérateur de troncature
9:     fin si
10:    si Taille de P̄_{t+1} < N̄ alors
11:        remplir P̄_{t+1} avec des individus dominés par P̄_t et P_t
12:    fin si
13:    Effectuer une sélection de tournoi binaire avec remplacement sur P̄_{t+1}
14:    t ← t + 1
15: fin tant que
16: Solutions ← P̄_t
```

Contrairement à la SPEA, SPEA2 utilise une stratégie (voir Algorithme 3) d'affectation de fitness à grain fin qui intègre les informations de densité. La taille de l'archive est fixée, c.-à-d, à chaque fois que le nombre d'individus non-dominés est inférieur à la taille de l'archive prédéfinie, l'archive est remplie par des individus dominés ; avec SPEA, la taille de l'archive peut varier au fil du temps. La technique de clustering, qui est invoquée lorsque le front non-dominé dépasse la limite d'archives, a été remplacée par une méthode alternative de troncature qui a des caractéristiques similaires mais ne fait pas perdre les points limites. Finalement, une autre différence à SPEA est que seulement les membres de l'archive qui participent au processus de sélection.

2.3.2.1 Attribution de fitness

Pour éviter la situation où les individus dominés par les mêmes membres de l'archive ont des valeurs identiques de fitness, SPEA2 prend en compte pour chaque individu les solutions dominantes et dominées. En particulier, chaque individu i dans l'archive $\overline{P}_t$ et

la population P_t à une valeur Strength $S(i)$ est attribuée qui représente le nombre de solutions qu'elle domine :

$$S(i) =\mid \{j \mid j \in P_t + \overline{P}_t \wedge i \succ j\} \mid \tag{2.1}$$

où $|.|$ désigne la cardinalitée d'un ensemble, $+$ est à l'union multi-ensembles et le symbole $\succ$ correspond à la relation de dominance de Pareto. Sur la base des valeurs de S, le fitness $R(i)$ d'un individu i est calculée comme suit :

$$R(i) = \sum_{j \in P_t + \overline{P}_t, j \succ i} S(j) \tag{2.2}$$

Par opposition à SPEA où seuls les membres de l'archive sont considérés dans le calcul de fitness, SPEA2 détermine le fitness par les $S(i)$ (Strength) de ses dominateurs dans les deux archives et la population [157]. Il est important de noter qu'ici le fitness doit être minimisée, c.-à-d, $R(i) = 0$ correspond à un individu non-dominé, alors que $R(i)$ avec une valeur élevée signifie que i est dominé par plusieurs individus (qui à son tour domine plusieurs individus). Figure 2.3 illustre ce schéma.

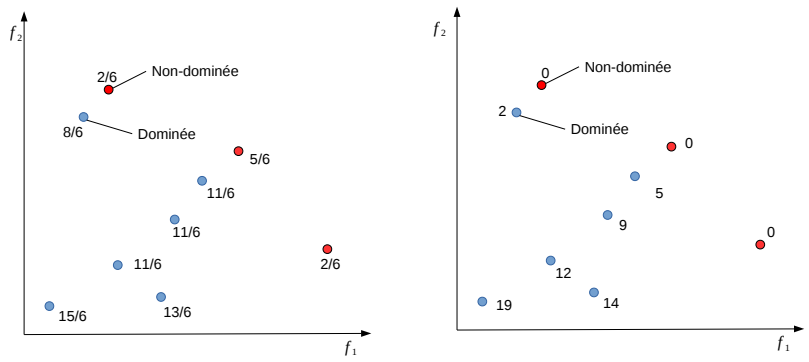


FIGURE 2.3: Comparaison des méthodes d'affectation de fitness dans SPEA et SPEA2 pour un problème de maximisation avec deux objectifs f_1 et f_2. Sur la gauche, les valeurs de fitness pour une population donnée selon le schéma de SPEA. Sur la droite, les valeurs de fitness R de SPEA2 pour la même population sont représentées.

Bien que l'affectation de fitness offre une sorte de mécanisme de Niching basée sur le concept de dominante de Pareto, il peut échouer lorsque la plupart des individus ne dominent pas d'autres. Par conséquent, une information de densité supplémentaire est incorporée pour la discriminer entre individus ayant des valeurs de fitness identiques. La technique d'estimation de densité utilisée dans SPEA2 est une adaptation de la méthode

du k-ième plus proche voisin, où la densité en un point quelconque est une fonction (décroissante) de la distance au k-ème point de données le plus proche. Ici, nous prenons simplement l'inverse de distance au k-ième voisin le plus proche comme estimation de densité. De façon plus précise, pour chaque individu i les distances (dans l'espace objectif) à tous les individus j dans l'archive et la population sont calculées et stockées dans une liste. Après le tri de la liste en l'ordre croissant, le k-ème élément donne la distance recherchée, notée σ_i^k. Comme une configuration commune, nous utilisons k égale à la racine carrée de la taille de l'échantillon [126], $k = \sqrt{N + \overline{N}}$. Par la suite, la densité $D(i)$ correspondant au i est définie par :

$$D(i) = \frac{1}{\sigma_i^k + 2} \tag{2.3}$$

Dans le dénominateur, on ajoute deux pour s'assurer que sa valeur est supérieure à zéro et que $D(i) < 1$. Enfin, l'ajout de $D(i)$ à la valeur de (Tableau) $R(i)$ d'un individu i donne son fitness $F(i)$:

$$F(i) = R(i) + D(i) \tag{2.4}$$

Le temps d'exécution de la procédure d'attribution de fitness est dominé par l'estimateur de densité ($\mathcal{O}(M^2 logM)$), tandis que le calcul des valeurs S et R est de complexité $\mathcal{O}(M^2)$ où $M = N + \overline{N}$.

2.3.2.2 Sélection

L'opération de mise à jour de l'archive (étapes 6 - 12 dans l'algorithme 1) dans SPEA2 diffère de celui de SPEA [157], la méthode de troncature empêche les solutions de limites d'être enlevées. Lors de la sélection, la première étape consiste à copier tous les individus non-dominés, c.-à-d, ceux qui ont un fitness inférieur à 1 à partir des archives et la population à l'archive de la prochaine génération :

$$\overline{P}_{t+1} = \{i | i \in P_t + \overline{P}_t \wedge F(i) < 1\} \tag{2.5}$$

Si le front non-dominé correspond exactement à l'archive ($|\overline{P}_{t+1}| = \overline{N}$) l'étape de sélection est terminée. Sinon, il peut y avoir deux situations : Soit l'archive est trop petite ($|\overline{P}_{t+1}| < \overline{N}$) ou trop grande ($|\overline{P}_{t+1}| > \overline{N}$). Dans le premier cas, les meilleurs $\overline{N} - |\overline{P}_{t+1}|$ individus dominés dans l'archive précédente et la population sont copiés dans la nouvelle

archive. Ceci peut être mis en œuvre par le tri de multi-ensemble $P_t + \overline{P}_t$ en fonction des valeurs de fitness et des copies des premiers $\overline{N} - |\overline{P}_{t+1}|$ individus i avec $F(i) \geq 1$ à partir de la liste ordonnée résultant de $\overline{P}_{t+1}$. Dans le second cas, lorsque la taille de multi-ensemble non-dominé dépasse $\overline{N}$, une procédure de troncature d'archives est invoquée qui élimine de manière itérative des individus de $\overline{P}_{t+1}$ jusqu'à ce que $|\overline{P}_{t+1}| = N$. Ici, à chaque itération l'individu i est choisi pour suppression pour laquelle $i \leq_d j$ pour tout $j \in \overline{P}_{t+1}$ avec

$$\begin{aligned} i \leq_d j \quad :\Leftrightarrow \quad & \forall 0 < k < |\overline{P}_{t+1}| : \sigma_i^k = \sigma_j^k \vee \\ & \exists 0 < k < |\overline{P}_{t+1}| : \left[\left(\forall 0 < l < k : \sigma_i^l = \sigma_j^l \right) \wedge \sigma_i^k < \sigma_j^k \right] \end{aligned} \tag{2.6}$$

où σ_i^k désigne la distance de i à son k-ième plus proche voisin dans $\overline{P}_{t+1}$. En d'autres termes, l'individu qui présente la distance minimale à un autre individu est choisi à chaque étape ; s'il y a plusieurs individus avec une distance minimale d'égalité est brisée en considérant la deuxième plus petite distance et ainsi de suite. Le fonctionnement la technique de troncature est illustré dans la Figure 2.4.

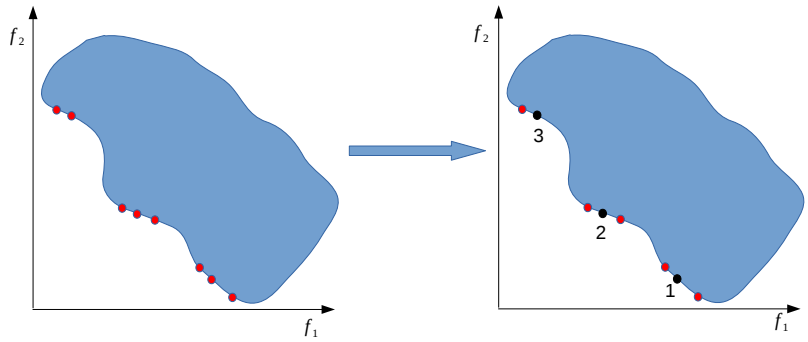


FIGURE 2.4: Illustration de la méthode de troncature d'archives utilisé dans SPEA2. Sur la droite, un ensemble non-dominé est affiché. À gauche, il décrit les solutions qui sont éliminés, et dans quel ordre, par l'opérateur de troncature (en supposant que $\overline{N} = 5$).

2.3.3 Algorithme évolutionnaire multobjectif basé sur la décomposition MOEA/D

MOEA/D (Multi-Objective Evolutionary Algorithm base on Decomposition)[152] décompose un problème d'optimisation à objectifs multiples en un nombre de sous-problèmes

d'optimisation scalaires et les optimise simultanément. Chaque sous-problème est optimisé en utilisant uniquement des informations de ses plusieurs sous-problèmes voisins, ce qui rend la complexité de calcul de MOEA/D inférieure à chaque génération que MOGLS et NSGA-II).

Il existe plusieurs approches pour convertir le problème de l'approximation de PF en un certain nombre de problèmes d'optimisation scalaires. Dans ce qui suit, nous présentons deux approches utilisées dans MOEA/D.

2.3.3.1 Décomposition par une approche de somme pondérée

Cette approche considère une combinaison convexe des différents objectifs. Soit $\lambda = (\lambda_1, \ldots, \lambda_m)^T$ un vecteur de poids, où $\lambda_i \geq 0$ pour tous $i = 1, \ldots, m$ et $\sum_{i=1}^{m} \lambda_i = 1$. Ensuite, la solution optimale au problème d'optimisation scalaire suivant :

$$\text{maximiser } g^{ws}(x|\lambda) = \sum_{i=1}^{m} \lambda_i f_i(x) \tag{2.7}$$

est un point Pareto optimale d'un problème multiobjectif (équation 1.1), où on utilise $g^{ws}(x|\lambda)$ pour souligner que λ est le vecteur de coefficient dans cette fonction objectif, tandis que x sont les variables à optimiser. Pour générer un ensemble de différents vecteurs de Pareto optimaux, on peut utiliser des différents vecteurs de poids λ dans le problème d'optimisation scalaire ci-dessus. Si le PF est concave (convexe dans le cas de la minimisation), cette approche pourrait bien fonctionner. Cependant, pas tous les vecteurs optimaux de Pareto peuvent être obtenus par cette approche dans le cas de PF non-concave. Pour surpasser ces insuffisances, certains efforts ont été faits pour incorporer d'autres techniques telles que ϵ-contrainte dans cette approche [14].

2.3.3.2 Décomposition par une approche Tchebycheff

Dans cette approche, le problème d'optimisation scalaire est dans la forme :

$$\text{minimiser } g^{te}(x|\lambda, \mathbf{z}^*) = \max_{1 \leq i \leq m} \{\lambda_i |f_i(x) - z_i^*|\} \tag{2.8}$$

où $\mathbf{z}^* = (z_1^*, \ldots, z_m^*)^T$ est le point de référence, c.-à-d, $z_i^* = \max\{f_i(x)\}$ pour $i = 1, \ldots, m$ (si on considère un problème de maximisation). Pour chaque point Pareto optimal x^* il existe un vecteur de poids λ tel que x^* est la solution optimale de 2.8 et chaque solution optimale de 2.8 est une solution optimale au sens de Pareto du pro-

blème défini par l'équation 1.1. Par conséquent, on est capable d'obtenir les différentes solutions Pareto optimales en modifiant le vecteur de poids.

2.3.3.3 Algorithme général de MOEA/D

L'algorithme évolutionnaire multiobjectif basé sur la décomposition (MOEA/D) doit décomposer le MOP considéré. Toute approche de décomposition peut servir cet objectif. Dans la description suivante, on suppose que l'approche Tchebycheff est employée.

Soit $\lambda^1, \ldots, \lambda^N$ un ensemble de vecteurs réparti uniformément et z^* un point de référence. Comme indiqué dans la section 2.3.3.2, le problème d'approximation de PF du problème 1.1 peut être décomposé en N sous-problèmes d'optimisation scalaires en utilisant l'approche Tchebycheff et la fonction objectif du j-ième sous-problème est :

$$\text{minimiser } g^{te}(x|\lambda^j, \mathbf{z}^*) = \max_{1 \leq i \leq m} \{\lambda_i^j | f_i(x) - z_i^* |\} \tag{2.9}$$

où $\lambda^j = (\lambda_1^j, \ldots, \lambda_m^j)^T$. MOEA/D minimise toutes ces N fonctions objectifs simultanément en une seule exécution.

Noter que g^{te} est continu de λ, la solution optimale de $g^{te}(x|\lambda^i, \mathbf{z}^*)$ doit être proche de celui de $g^{te}(x|\lambda^j, \mathbf{z}^*)$ si λ^i et proche à λ^j. Par conséquent, toute information sur ceux de g^{te} avec des vecteurs de poids proche à λ^i va être utile pour optimiser $g^{te}(x|\lambda^i, z^*)$.

Dans MOEA/D, un voisin de vecteur de poids λ^i et défini comme un ensemble de vecteurs de poids les plus proches à lui dans $\{\lambda_1, \ldots, \lambda_N\}$. Le voisinage du i-ème sous-problème comprend tous les sous-problèmes avec les vecteurs de poids voisins de λ^i. La population est composée de la meilleure solution trouvée jusqu'à maintenant pour chaque sous-problème. Seules les solutions actuelles pour ses sous-problèmes voisins sont exploitées pour l'optimisation d'un sous-problème en MOEA/D.

À chaque génération t, MOEA/D avec l'approche Tchebycheff maintient :

- Une population de N points $x^1, \ldots, x^N$, où x^i est la solution courante pour le i-ème sous-problème.
- $FV^1, \ldots, FV^N$, où FV^i est la valeur F de x^i, c.-à-d, $FV^i = F(x^i)$ pour chaque $i = 1, \cdots, N$.
- $z = (z_1, \cdots, z_m)^T$, où z_i est la meilleure valeur trouvée jusqu'à présent pour l'objectif f_i.
- Une population externe (EP), qui est utilisée pour stocker des solutions non-dominées trouvées lors de la recherche.

L'algorithme fonctionne comme suit :

Algorithme 4 Pseudocode de l'algorithme MOEA/D

1: Générer aléatoirement le vecteur de poids $(\lambda^t, \cdots, \lambda^N)$
2: EP = $\emptyset$
3: Déterminer les T solutions adjacentes les plus proches
4: $B(i) = \{i_1, \cdots, i_T\}$ avec $\lambda^{i_1}, \cdots, \lambda^{i_T}$ sont les T vecteurs de poids plus proches à λ^i pour chaque $i = 1, \cdots, N$.
5: Générer une population initiale $x^1, \cdots, x^N$
6: $FV^i = F(x^i)$
7: Initialiser $z = (z_1, \cdots, z_m)^{Trans}$
8: **pour** $i = 1, \cdots, N$ **faire**
9: Choisir k, l de $B(i)$ et produire y depuis x^k et x^l
10: Réparer y pour produire y'
11: **pour** $j = 1, \cdots, m$ **faire**
12: **si** $z_j < f_j(y')$ **alors**
13: $z_j = f_j(y')$
14: **fin si**
15: **fin pour**
16: **pour** $j \in B(i)$ **faire**
17: **si** $g^{te}(y'|\lambda^j, z) \leq g^{te}(x^j|\lambda^j, z)$ **alors**
18: $x^j = y'$
19: $FV^j = F(y')$
20: **fin si**
21: **fin pour**
22: Mettre à jour EP
23: Supprimer de EP tous vecteurs dominés par $F(y')$
24: Ajouter $F(y')$ à EP si aucuns vecteurs dans EP domine $F(y')$
25: **fin pour**
26: Si le critère d'arrêt non satisfait goto 8
27: Retourner EP

Dans l'initialisation, $B(i)$ contient les indices de T vecteurs proche de λ^i. Nous utilisons la distance euclidienne pour mesurer la proximité entre deux vecteurs de pondération. Par conséquent, le vecteur le plus proche à λ^i est lui-même, et ensuite $i \in B(i)$. Si $j \in B(i)$ le j-ème sous-problème peut être considéré comme un voisin de la i-ème

sous-problème[132].

Dans la i-ème passe de la boucle à l'instruction 8, les T sous-problèmes voisins du i-ème sous-problème voisin sont considérés. Depuis, x^k et x^l à l'étape 9 sont les meilleures solutions actuelles aux voisins de la i-ème sous-problème, leur descendant y devrait, être une bonne solution pour le i-ème sous-problème. Dans l'étape 10, une heuristique spécifique au problème est utilisée pour réparer y, et/ou optimiser la i-ème g^{te}. Par conséquent, la solution résultante y' est réalisable et très susceptibles d'avoir une valeur de fonctions inférieure pour les voisins du i-ème sous-problème. L'étape 19 considère tous les voisins du i-ème sous-problème, il remplace x^j avec y' si y' est plus performant que x^j à l'égard du j-ème sous-problème. FV^j est nécessaire dans le calcul de la valeur de $g^{te}(x^j|\lambda^j, z)$ dans l'étape 19.

Trouver le point de référence exacte z^* souvent prend beaucoup de temps, nous utilisons z, qui est initialisé à l'étape 7 par une méthode spécifique au problème et mise à jour à l'étape 13, comme un substitut pour z^* dans g^{te}. La population externe EP, initialisé dans l'étape 2, est mise à jour par la nouvelle solution de y' générées à l'étape 22.

Conclusion

Dans ce chapitre, une revue de littérature des MOEAs a été présentée. L'utilisation des algorithmes évolutionnaires dans l'optimisation multiobjectif a donné preuve d'une grande réussite. Les algorithmes évolutionnaires ont la capacité de trouver des meilleures approximations de solutions Pareto. Les algorithmes évolutionnaires comportent deux étapes ; la sélection et la reproduction. Ces algorithmes se diffèrent l'un de l'autre dans le processus de sélection des meilleurs aussi dans la manière de reproduire les nouveaux individus. Par exemples l'algorithme NSGA-II utilise les opérateurs classiques de reproduction qui sont le croisement et la mutation, même chose pour l'algorithme SPEA2, alors que la différence réside dans la sélection et la manière d'affectation de fitness.

Dans ce chapitre on a illustré aussi, que les MOEAs sont classés en trois catégories principales qui sont les MOEAs avec un framework basé préférence, un framework basé dominance, et un framework basé décomposition. Plusieurs algorithmes de l'état de l'art dans différents cadres de l'optimisation multiobjectifs ont été discutés. En outre, les EDA pour l'optimisation multiobjectif ont été soulignés. Dans ce chapitre on a décrit aussi les algorithmes principaux qui vont être utilisés dans cette thèse.

Chapitre 3

Algorithmes à Estimation de Distribution

Introduction

Les Algorithmes à Estimation de distribution (Estimation of Distribution Algoritm, EDA) [91], également appelés algorithmes génétiques à construction de modèle probabilistes(Probabilistic Model-Building Genetic Algorithm ,PMBGA) et algorithmes évolutionnaires à estimation de densité itéré (Iterated Density Estimation Evolutionary Algorithm, IDEA), l'algorithme vue l'optimisation comme une série de mises à jour incrémentielles d'un modèle probabiliste, en commençant par le modèle codant une distribution uniforme des solutions admissibles et finissant avec le modèle qui génère uniquement l'optimum global. Dans les dernières années, EDAs ont été appliquées à de nombreux problèmes d'optimisation [7, 8, 10, 56, 118]. Dans plusieurs de ces études, EDAs ont été montrés qu'ils peuvent résoudre les problèmes qui étaient intraitables avec d'autres techniques ou aucune autre technique pourrait atteindre des résultats comparables [12, 64].

Toutefois, le motif de l'utilisation des EDAs dans la pratique est non seulement que ces algorithmes peuvent résoudre des problèmes d'optimisation difficiles, mais que, en plus de l'optimum ou son approximation, EDAs offrent un modèle informatique compacte du problème représenté par une série de modèles probabilistes [105]. Ce modèle probabiliste révèle beaucoup d'informations sur le domaine du problème, qui peut à son tour être utilisé pour l'optimisation des problèmes similaires. Alors que plusieurs métaheuristiques essentiellement utilisent des distributions de probabilités implicites en utilisant une combinaison d'opérateurs de recherche stochastiques, avoir une présentation de so-

lutions du problème avec une série de modèles probabilistes explicites donne EDA un avantage sur la plupart des autres métaheuristiques.

Ce chapitre fournit une introduction aux EDAs. En outre, le chapitre présente de nombreux concepts pour obtenir des informations supplémentaires sur cette classe d'algorithmes.

3.1 Procédure principale d'un EDA

3.1.1 Définition de problème

Un problème d'optimisation peut être défini par (1) un ensemble de solutions possibles au problème et (2) une procédure pour évaluer la qualité de ces solutions. L'ensemble des solutions possibles est souvent défini en utilisant une représentation générale de solutions admissibles et un ensemble de contraintes. La procédure d'évaluation de la qualité des solutions peut être définie soit comme une fonction qui doit être minimisée ou maximisée (souvent désignée comme une fonction objectif ou de fitness). La tâche est de trouver une solution à partir de l'ensemble des solutions potentielles qui maximise la qualité telle que définie par la procédure d'évaluation [116].

Comme un exemple, considérons le problème de satisfiabilité maximale pour les formules de logique propositionnelle définies dans la forme normale conjonctive avec 3 littéraux par clause (MAX3SAT). Dans MAX3SAT, chaque solution potentielle est définie d'une interprétation de propositions.La qualité d'une solution est mesurée par le nombre de clauses qui sont satisfaites par l'interprétation spécifique. La tâche est de trouver une interprétation qui maximise le nombre de clauses satisfaites.

Sans des hypothèses supplémentaires sur le problème, une façon de trouver l'optimum est de répéter trois étapes principales :

— Générer des solutions candidates.
— Évaluer les solutions générées.
— Mettre à jour la procédure pour générer de nouvelles solutions candidates selon les résultats de l'évaluation.

L'idéal serait que l'exécution de ces trois étapes générerait l'optimum global ou son approximation. Des différents algorithmes cherchent à mettre en œuvre les trois étapes ci-dessus de différentes façons, mais l'idée principale reste la même ; de manière itérative mettre à jour la procédure qui génére les solutions afin que ces solutions générées

continuent à s'améliorer en termes de qualité.

3.1.2 Procédure EDA

Dans les EDAs, l'idée principale est de maintenir un modèle probabiliste explicite qui représente une distribution de probabilité sur les solutions. Dans chaque itération, le modèle est ajusté en fonction des résultats de l'évaluation de solutions trouvées, l'ajustement a pour but d'obtenir des solutions meilleures dans les itérations suivantes. Notez que l'utilisation d'un modèle probabiliste explicite rend l'EDA très différent de nombreuses autres métaheuristiques, tels que les algorithmes génétiques [23, 55, 66] ou le recuit simulé [25, 77], dans lequel la distribution de probabilité utilisée pour générer de nouvelles solutions est souvent définie implicitement par un opérateur de recherche ou une combinaison de plusieurs opérateurs de recherche. Les chercheurs distinguent souvent deux types principaux des EDAs :

— Les EDAs basés population. EDAs basés population maintiennent une population (multiset) de solutions commençant par une population générée au hasard selon une loi uniforme sur toutes les solutions admissibles. Chaque itération commence par la création d'une population de solutions candidates à l'aide de l'opérateur de sélection, ce qui donne la préférence aux solutions de meilleure qualité. Toute méthode de sélection connue dans les algorithmes évolutionnaires peut être utilisée. Un modèle probabiliste est ensuite construit pour les solutions choisies. De nouvelles solutions sont créés en échantillonnant la distribution codée par le modèle construit. Les nouvelles solutions sont ensuite incorporées dans la population d'origine en utilisant un opérateur de remplacement. En remplacement complet, par exemple, l'ensemble de la population d'origine de solutions candidates est remplacé par les nouveaux. Un pseudo-code d'un EDA basé sur la population est représenté dans l'algorithme 5.

Algorithme 5 Pseudocode d'un EDA basé population

```
t ← 0
Générer une population initiale P(t) de solutions aléatoires
tant que le critère n'est pas satisfait faire
    Évaluer toutes les solutions dans P(t)
    Choisir parmi P(t) un groupe de parents S(t) préférant les meilleurs
    Construire un modèle probabiliste M(t) de S(t)
    Générer des nouveaux solutions O(t) en échantillonnant M(t)
    Construire P(t+1) en combinant O(t) et P(t)
    t ← t + 1
fin tant que
```

— Les EDAs incrémenteaux. Dans les EDAs incrémenteaux, la population de solutions candidates est entièrement remplacée par un modèle probabiliste. Le modèle est initialisé de sorte qu'il code pour la distribution uniforme toutes les solutions admissibles. Le modèle est ensuite mis à jour de façon incrémentale en répétant le processus de (1) l'échantillonnage de plusieurs solutions candidates du modèle actuel et (2) l'ajustement du modèle basé sur l'évaluation de ces solutions candidates et leur comparaison de sorte que le modèle devient plus susceptible de générer des solutions de haute qualité dans les itérations suivantes. Un pseudo-code d'un EDA incrémental est représenté dans l'algorithme 6.

Algorithme 6 Pseudocode d'un EDA incrémental

```
t ← 0
Initialiser le modèle M(0) avec une distribution uniforme
tant que le critère n'est pas satisfait faire
    Générer la population P(t) avec l'échantillonnage de M(t)
    Évaluer toutes les solutions dans P(t)
    Créer un modèle M(t+1) en ajustant M(t) selon P(t) évalué
    t ← t + 1
fin tant que
```

Les EDAs Incrémenteaux génèrent souvent seulement quelques solutions candidates à un moment, alors que les EDAs basé-population travaillent souvent avec une grande population de solutions candidates. Néanmoins, il est facile de constater que les deux approches sont essentiellement les mêmes, parce que même les EDAs basé-population peuvent être reformulés de manière incrémentale.

Les principales composantes d'un EDA basé-population comprennent donc :

1. Un opérateur de sélection pour sélectionner des solutions prometteuses.
2. Une classe de modèles probabilistes à utiliser pour la modélisation et de l'échantillonnage.
3. Une procédure pour l'apprentissage d'un modèle probabiliste pour les solutions choisies.
4. Une procédure pour échantillonner le modèle probabiliste construit.
5. Un opérateur de remplacement pour combiner les anciennes et nouvelles populations de solutions candidates.

Les principales composantes d'un EDA incrémental comprennent donc :

1. Une classe de modèles probabilistes.
2. Une procédure pour ajuster le modèle probabiliste basé sur les nouvelles solutions et de leurs évaluations.
3. Une procédure pour échantillonner le modèle probabiliste.

Le schéma général d'un EDA basé-population est assez similaire à celui d'un algorithme évolutionnaire traditionnel (EA) [43] ; guider la recherche vers des solutions prometteuses en effectuant itérativement la sélection et la variation. En particulier, les composants (1) et (5) sont précisément les mêmes que ceux utilisés dans d'autres algorithmes évolutionnaires. Les composants (2), (3) et (4), cependant, sont uniques pour les algorithmes à estimation de distribution (EDA), et constituent leur moyen de produire des variations, par opposition à l'utilisation des opérateurs de croisement et mutation comme c'est souvent le cas avec d'autres EAs.

Il faut noter que cette perspective ouvre une voie à la conception des procédures de recherche en apportant au domaine des algorithmes évolutionnaires une énorme quantité de connaissances sur l'apprentissage automatique, et en particulier de modèles graphiques probabilistes. L'idée clé d'un EDA est de voir une population de bonnes solutions précédemment visitées en tant que données, apprendre un modèle de ces données, et utiliser le modèle résultant pour déduire d'autres solutions qui pourraient être bonnes. Cette approche est puissante, permettant à un algorithme de recherche d'apprendre et de s'adapter par rapport au problème d'optimisation en cours de résolution.

3.1.3 Simulation d'un exemple de EDA

Pour mieux comprendre la procédure EDA, cette section présente une simulation d'un EDA simple. Le but de la présentation de la simulation est de clarifier les éléments de procédure de base d'un EDA et de construire une intuition sur les dynamiques d'une exécution d'un EDA.

La simulation suppose que les solutions candidates sont représentées par des chaînes binaires de longueur fixe $n > 0$. La fonction objectif est de maximiser ONEMAX, qui est définie comme la somme des bits dans la chaîne binaire d'entrée $(X_1, X_2, \ldots, X_n)$

$$f_{ONEMAX}(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} X_i \tag{3.1}$$

La qualité d'une solution augmente avec le nombre de 1 dans la chaîne d'entrée, et l'optimum correspond à la chaîne de tous les 1.

Pour modéliser et échantillonner des solutions candidates, la simulation utilise un vecteur de probabilité [117]. Un vecteur de probabilité p de chaînes binaires de n-bits a n composantes, $p = (p_1, p_2, \ldots, p_n)$. La composante p_i représente la probabilité d'observer un 1 en position i d'une chaîne de solution. Pour apprendre le vecteur de probabilité, p_i est mis à la proportion de 1 dans la position i observée dans l'ensemble des solutions sélectionnées. Pour échantillonner une nouvelle solution $(X_1, X_2, \ldots, X_n)$, les composantes du vecteur de probabilité sont interrogées et chaque X_i est mis à 1 avec une probabilité p_i, et à 0 avec une probabilité de $1 - p_i$.

Le résultat attendu de l'apprentissage et de l'échantillonnage du vecteur de probabilité est que la population des solutions choisies et de la population de nouvelles solutions ont la même proportion de 1 dans chaque position. Cependant, comme l'échantillonnage considère chaque nouvelle solution indépendamment des autres candidates, les proportions réelles peuvent varier un peu de leurs valeurs attendues. L'algorithme à estimation de distribution à vecteur de probabilité décrit ci-dessus est généralement considéré comme "Univariate Marginal Distribution Algorithm" (UMDA) [99] ; d'autres EDA sur la base du modèle de vecteur de probabilité seront discutés dans la Section 3.3.1.

Pour avoir une simulation simple , nous considérons un 5-bits ONEMAX, une population de taille $N = 6$, et la sélection avec un seuil $\tau = 50\%$. Rappelons que la sélection avec $\tau = 50\%$ sélectionne la moitié supérieure de la population actuelle.

La Figure 3.1 montre les deux premières itérations de la simulation EDA. La population initiale est générée d'une manière aléatoire. On sélectionne ensuite le meilleur

50% des solutions en fonction de leur évaluation à l'aide de ONEMAX pour former l'ensemble des solutions prometteuses. Ensuite, le vecteur de probabilité est créé sur la base des solutions retenues, et la distribution codée par le vecteur de probabilité est échantillonné pour générer de nouvelles solutions. La population produite remplace la population d'origine et la procédure se répète.

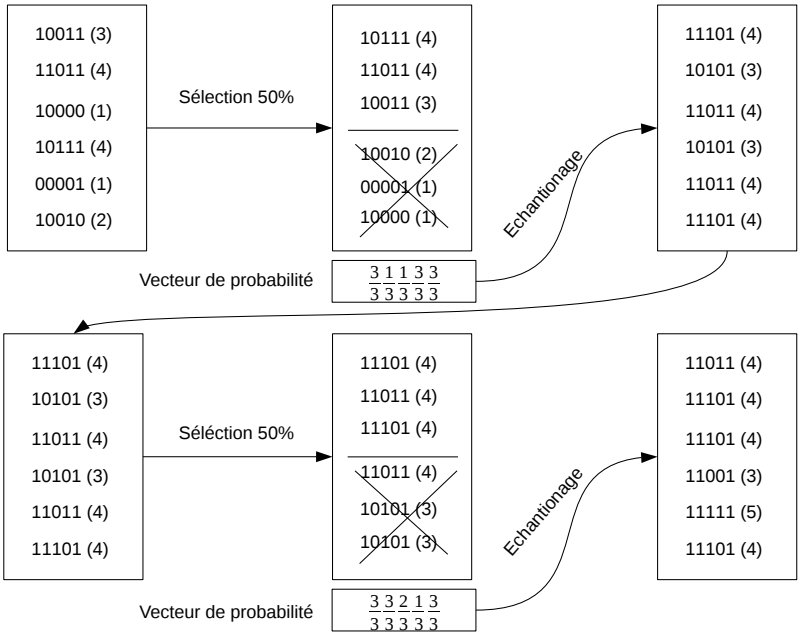


FIGURE 3.1: Simulation simple d'un EDA basé sur le modèle probabiliste à vecteur de problème ONEMAX. Les valeurs de fitness des solutions sont présentées à l'intérieur de parenthèses.

Dans les deux itérations de la simulation, la valeur de fonction objectif moyenne dans la nouvelle population est supérieure à la valeur moyenne de la population avant la sélection. L'augmentation de la qualité moyenne de la population est de bonnes nouvelles pour nous parce que nous voulons maximiser la fonction objectif, mais pourquoi est-ce possible ? Car pour ONEMAX les solutions avec plus de 1s sont meilleures que celles avec moins de 1s, la sélection devrait augmenter le nombre de 1 dans la population. On ne prévoit pas l'apprentissage et l'échantillonnage du vecteur de probabilité de créer ou de détruire tous les bits et qui est la raison pour laquelle la nouvelle population de solutions candidates devrait contenir plus de 1s que la population d'origine (à la fois dans la proportion et dans le nombre réel). Parce que la valeur de ONEMAX augmente avec le nombre de 1, nous pouvons nous attendre de la qualité globale de la population d'augmenter au fil du temps. Idéalement, chaque itération devrait augmenter les valeurs de fonction objectif de la population.

Néanmoins, l'augmentation de la valeur moyenne de la fonction objectif ne montre que la moitié du meilleur solution. Une augmentation similaire de la qualité de la population dans la première itération pourrait être obtenue en répétant simplement la sélection seule sans l'utilisation du modèle probabiliste. Cependant, en appliquant la sélection seule, aucune nouvelle solution ne sera créée et l'algorithme résultant ne produit aucune variation du tout (c.-à-d, il n'y a pas l'exploration de nouvelles solutions). Puisque la population initiale est générée au hasard, l'EDA avec la sélection seule serait juste un mauvais algorithme pour l'obtention des meilleures solutions depuis la population initiale. L'apprentissage et l'échantillonnage du modèle probabiliste fournissent un mécanisme qui permet à la fois (1). l'amélioration de la qualité des nouvelles solutions (sous certaines hypothèses), et (2)la facilité l'exploration de l'ensemble des solutions admissibles.

Ce que nous avons vu dans cette simulation était un exemple de la forme la plus simple des EDAs. La classe présumée de modèles probabilistes, le vecteur de probabilité, a une structure fixe. Dans ces circonstances, la procédure pour l'apprentissage, il devient trivial car il n'y a vraiment pas d'autres modèles à choisir. Cette classe de EDA est très limitée dans ce qu'elle peut faire. Comme nous le verrons dans la suite, il y a d'autres catégories de EDAs qui ont des modèles probabilistes plus riches capables de capturer les interactions entre les variables d'un problème donné. Plus important encore, ces interactions peuvent être apprises automatiquement sur un problème en fonction du problème. Il en résulte bien sûr, à une procédure de construction d'un modèle plus complexe, mais l'effort supplémentaire a été montré pour être bien utile, surtout lors de la résolution des problèmes d'optimisation plus difficiles [34, 82, 97].

3.2 Classification des modèles EDA

Cette section fournit un aperçu de haut niveau des caractéristiques distinctives des modèles probabilistes. Les caractéristiques sont discutées par rapport à (1) les types d'interactions couverts par le modèle et (2) les types de distributions locales. Cette section se concentre uniquement sur les principales caractéristiques des modèles probabilistes ; un aperçu plus détaillé des EDAs pour les différentes représentations de solutions candidates sera couvertes dans les sections suivantes.

3.2.1 Classification basée sur la décomposition du problème

Pour faire de l'estimation et de l'échantillonnage traitable avec des échantillons de taille raisonnable, la plupart des EDAs utilisent des modèles probabilistes qui décomposent le problème en utilisant l'indépendance inconditionnelle ou conditionnelle. La façon dont un modèle décompose le problème fournit une caractéristique importante qui distingue les différentes classes de modèles probabilistes.

La plupart des EDAs supposent que les solutions candidates sont représentées par des vecteurs de longueur fixe de variables et ils utilisent des modèles graphiques pour représenter la structure du problème sous-jacent. Les modèles graphiques permettent aux praticiens de représenter les deux dépendances directes entre les variables de problèmes ainsi que les hypothèses d'indépendance. Une façon de classer les modèles graphiques est d'envisager une hiérarchie des types de modèles basés sur la complexité d'un modèle (voir la figure 3.2 pour des exemples.) [49] :

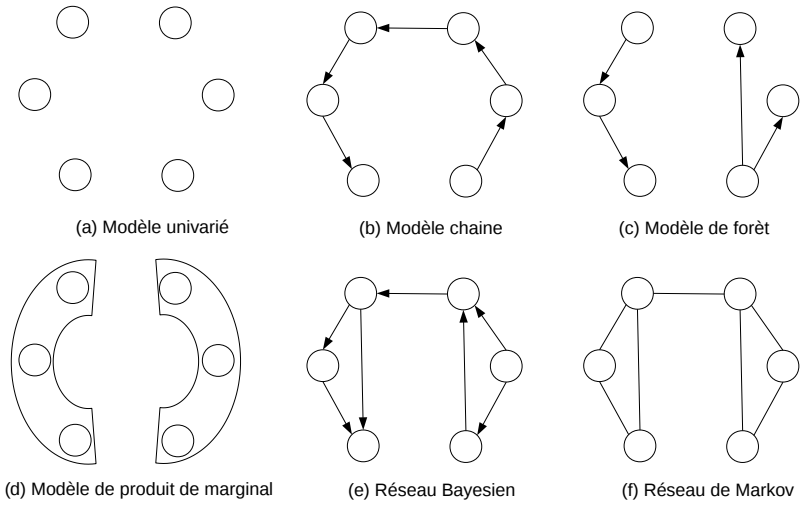


FIGURE 3.2: Des exemples illustratifs de modèles graphiques.les variables du problème sont affichées sous forme de cercles et les dépendances sont présentées comme des arêtes entre les variables ou groupes de variables. (a) modèle univarié. (b) modèle chaîne. (c) modèle de foret. (D) modèle de produit marginal. (E) réseau bayésien. (F) réseau de Markov.

— *Aucune dépendance.* Dans les modèles qui assument l'indépendance totale, chaque variable est supposée être indépendante de toute autre variable. Autrement dit, la distribution de probabilité $P(X_1, X_2, \ldots, X_n)$ du vecteur $(X_1, X_2, \ldots, X_n)$ de n

variables est un produit de la répartition des variables d'individus.

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i) \tag{3.2}$$

La simulation présentée dans la section 3.1.3 a été basée sur un modèle qui a supposé l'entière indépendance de variables du problème binaires.Les EDAs basées sur des modèles univariés qui assument l'indépendance totale de variables du problème comprend Equilibrium Genetic Algorithm (EGA), Population-Based Incremental Learning (PBIL), UMDA [99], Compact Genetic Algorithm(CGA) [60], le Hill Climbing stochastique avec l'apprentissage par des vecteurs de distributions normales [113], et PBIL continue [122].

— *Dépendances en paires.* Dans cette classe de modèles, les dépendances entre les variables forment un graphique de l'arbre ou forêt de graphe. Dans un graphe d'arbre, chacune des variables à l'exception de la racine, est conditionnée par ses parents dans un arbre qui contient toutes les variables. Un graphe de forêt, est une collection d'arbres déconnectés. Encore une fois, la forêt contient toutes les variables du problème. En désignant par R l'ensemble des racines des arbres dans une forêt, et par $X = (X_1, X_2, \ldots, X_n)$ l'ensemble du vecteur de variables, la distribution de cette classe peut être exprimée comme :

$$P(X_1, X_2, \ldots, X_n) = \prod_{X_i \in R} P(X_i) \times \prod_{X_i \in X \setminus R} P(X_i \mid parent(X_i)) \tag{3.3}$$

Un type particulier d'un modèle d'arbre est parfois distingué, dans laquelle les variables forment une séquence (ou une chaîne), et chacune des variables à l'exception de la première dépend directement de son prédécesseur. En désignant par $\pi(i)$ l'indice de la i-ième variable de la séquence, la distribution est donnée par :

$$P(X_1, X_2, \ldots, X_n) = P(X_{\pi(1)}) \times \prod_{i=2}^{n} P(X_{\pi(i)} \mid X_{\pi(i-1)}) \tag{3.4}$$

EDAs basées sur des modèles avec des dépendances paires comprennent Mutual Information Maximizing Input Clustering (MIMIC) [34], EDA basé sur des arbres de dépendance, et Bivariate Marginal Distribution Algorithm (BMDA) [108].

— *Dépendances multivariées.* Les modèles multivariés représentent les dépendances en utilisant soit des graphes acycliques orientés ou graphes non orientés. Deux modèles représentatifs sont populaires dans EDAs : (1) les réseaux bayésiens et (2)

les réseaux de Markov. Un réseau bayésien est représenté par un graphe acyclique orienté où chaque nœud correspond à une variable et chaque arête définit une dépendance conditionnelle directe. La distribution de probabilité codée par un réseau bayésien peut être écrite comme :

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid parents(X_i)) \tag{3.5}$$

Un réseau bayésien représente la décomposition de problème par les hypothèses de l'indépendance conditionnelle ; chaque variable est supposée être indépendante de tous ses antécédents dans l'ordre ancestral des variables, compte tenu des valeurs des parents de la variable. Notez que tous les modèles examinés jusqu'ici étaient des cas particuliers de réseaux bayésiens. En fait, un réseau bayésien peut représenter une distribution multivariée arbitraire. Toutefois, pour qu'un tel modèle soit pratique, il est souvent souhaitable d'envisager des réseaux bayésiens de complexité limitée.

Dans les réseaux de Markov, deux variables sont supposées être indépendantes les unes des autres, étant donné un sous-ensemble de variables définissant la condition, si tous les chemins entre ces variables sont séparées par une ou plusieurs variables de la condition. Une sous-classe particulière des modèles multivariés est parfois considérée dans laquelle les variables sont divisées en groupes disjoints, qui sont indépendants les uns des autres. Ces modèles sont appelés modèles marginaux de produits (Marginal Product Models, MPM).

Les EDAs basés sur des modèles avec des dépendances multivariées comprennent Factorized Distribution Algorithm (FDA) , Learning FDA (LFDA) [98], Estimation of Bayesian Network Algorithm (EBNA), Bayesian Optimization Algorithme (BOA) [108] et sa version hiérarchique (hBOA) [108], Extended Compact Genetic Algorithm (ECGA) [60].

— *Dépendance Totale.* Les modèles peuvent être utilisés à qui ne font pas d'hypothèses d'indépendance. Toutefois, ces modèles doivent généralement imposer un certain nombre d'autres restrictions sur la distribution afin d'assurer que les modèles restent flexibles pour un nombre élevé de variables.

Il existe deux types supplémentaires de modèles probabilistes qui ont été utilisés dans les EDAs et qui fournissent un mécanisme un peu différent pour décomposer le problème :

— *Modèles de grammaire.* Certaines EDAs utilisent des grammaires stochastiques ou

déterministes pour représenter la distribution de probabilité sur les solutions candidates. L'avantage de grammaires est qu'ils permettent la modélisation des structures à longueur variable. Pour cette raison, les distributions de grammaire sont principalement utilisées comme base pour la mise en œuvre de la programmation génétique en utilisant EDA [96], ce qui représente des solutions candidates en utilisant des arbres marqués de taille variable. Les modèles de grammaire sont utilisés, par exemple, dans les EDA à base de grammaire probabiliste pour la programmation génétique [17],les EDAs basés sur les grammaires probabilistes avec annotations latentes [62].

— *Modèles basées sur les caractéristiques.* Les modèles basées sur les caractéristiques codent la distribution du voisinage d'une solution candidate en utilisant des sous-structures de position indépendante, qui peuvent être trouvées dans une variété de positions dans des solutions de longueur fixe ou de longueur variable. Cette approche est utilisée dans le BOA basé sur les caractéristiques [95]. D'autres caractéristiques peuvent être découvertes, codées, et utilisées pour guider l'exploration de l'espace des solutions.

3.2.2 Classification basée sur les distributions locales dans les modèles graphiques

Indépendamment de la façon dont un modèle graphique décompose le problème, chaque modèle doit également assumer une ou plusieurs classes de distributions pour encoder les distributions conditionnelles et marginales locales. Certaines des classes les plus courantes de distributions locales sont discutées ci-dessous :

— *Tables de probabilités.* Pour les représentations discrètes, les probabilités conditionnelles et marginales peuvent être encodées en utilisant des tables de probabilités, qui définissent une probabilité pour chaque combinaison pertinente de valeurs dans chaque terme de probabilité conditionnelle ou marginale. Par exemple, dans la simulation dans la section 3.1.3, dans laquelle la distribution de probabilité pour chaque position de la chaîne i est représentée par la probabilité p_i d'un 1 ; la probabilité d'un 0 dans la même position était simplement $1 - p_i$. Comme autre exemple, dans les réseaux bayésiens, pour chaque variable, une table de probabilité peut être utilisée pour définir les probabilités conditionnelles, de toute valeur de la variable donnée, toute combinaison des valeurs des parents de la variable. Alors que les tables de probabilités ne peuvent pas représenter directement des distributions de probabilités continues, elles peuvent être utilisées même pour les représentations

à valeurs réelles en combinaison avec une méthode de discrétisation qui mappe des variables à valeurs réelles en catégories distinctes ; chacune des catégories distinctes peut alors être représentée en utilisant une seule entrée de probabilité. Les tables de probabilités sont utilisées, par exemple, dans UMDA, BOA et ECGA. Un exemple de table de probabilités conditionnelle est représentée sur la Figure 3.3.

— *Les arbres de décision.* Pour éviter de trop grandes tables de probabilités quand beaucoup de probabilités sont similaires ou négligeable, les structures locales plus avancées tels que des arbres de décision, graphes de décision peuvent être utilisés. Dans les arbres de décision, par exemple, les probabilités sont stockées dans des feuilles d'un arbre de décision dans lequel chaque nœud interne représente un test sur une variable et les fils du nœud correspondent aux différents résultats de l'essai. Les arbres de décision et des graphes de décision peuvent également être utilisés en combinaison avec des variables à valeurs réelles, dans lesquelles les feuilles stockent une distribution continue en quelque sorte. Des structures plus avancées tels que les arbres de décision et des graphes de décision sont utilisés, par exemple, dans décision-graphe BOA (dBOA), hiérarchique BOA (hBOA) [106], et mixte BOA (MBOA). Un exemple arbre de décision pour représenter des probabilités conditionnelles est représenté sur la Figure 3.3.

X2	X3	X4	P(X1|X2, X3, X4)
0	0	0	0.75
0	0	0	0.25
0	0	0	0.25
0	0	0	0.25
1	1	1	0.20
1	1	1	0.20
1	1	1	0.20
1	1	1	0.20

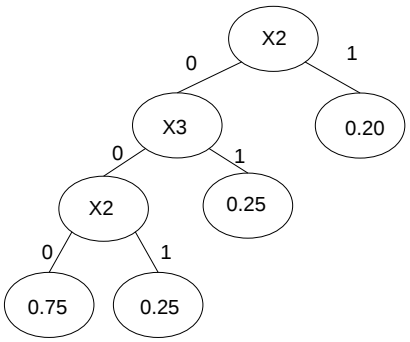


FIGURE 3.3: Une table de probabilité conditionnelle pour $p(X_1 \mid X_2, X_2, X_3, X_4)$ et un arbre de décision correspondant, qui réduit le nombre de paramètres (probabilités) de 8 à 4

— *Distributions continues Multivariées.* La distribution normale est la distribution la plus utilisée dans EDA pour représenter les distributions univariées ou multivariées de variables à valeurs réelles. Une distribution normale multivariée peut coder une corrélation linéaire entre les variables en utilisant la matrice de covariance, mais elle est souvent inefficace dans la représentation de nombreux autres types d'interactions [16, 102]. Les distributions normales ont été utilisées dans de nombreuses EDA pour les vecteurs de valeur réelle [18, 81], bien que dans de nom-

breuses EDA à valeurs réelles des distributions plus avancées ont été utilisées.

— *Mixture de distributions.* Une mixture de distribution est constituée de plusieurs composants. Chaque composant est représenté par un modèle probabiliste local spécifique. Les mixtures de distributions ont été utilisées dans EDA en particulier pour permettre aux EDAs des représentations à valeurs réelles pour faire face à des distributions de valeurs réelles dans lequel une distribution d'une seule pointe ne suffit pas. Les Mixtures de Distributions ont été utilisées, par exemple, dans les algorithmes itératifs à valeurs réelles d'estimation de densité ou BOA réel codé [4]. L'utilisation de mixture de distributions est plus populaire dans EDA pour les représentations à valeurs réelles. Bien que les mixtures de distributions aient également été utilisées pour représenter les distributions sur les représentations distinctes dans lesquelles la population se compose de plusieurs clusters dissemblables et EDAs multiobjectif [19, 111].

— *Histogrammes.* Dans certain nombre de EDAs de représentations à valeurs réelles pour encoder les distributions locales, des variables à valeurs réelles ou des ensembles de telles variables sont divisés en régions rectangulaires en utilisant un modèle d'histogramme et un modèle probabiliste distincte est utilisé pour représenter la distribution dans chaque région. Les modèles d'histogramme peuvent être considérés comme une sous-classe particulière des modèles d'arbres de décision pour les variables à valeurs réelles. Dans EDAs à valeurs réelles, les histogrammes ont été utilisés, par exemple, Histogram Population-Based Incremental Learning (HPBIL) [135].

3.3 Aperçu sur les EDAs

Cette section donne un aperçu des EDAs basés sur la représentation de solutions ; Bien que certaines des EDAs peut être utilisés dans plusieurs représentations. En raison du grand volume de travail dans EDAs dans les deux dernières décennies, nous ne visons pas d'énumérer toutes les variantes des EDAs. Par contre, nous nous concentrons sur certains représentants considéré les plus importants.

3.3.1 EDAs pour des chaines de taille fixe sur alphabets finis

Les EDAs pour les solutions représentées par des chaînes de longueur fixe sur un alphabet fini peuvent utiliser une variété de types de modèles ; des modèles univariés

simples aux réseaux bayésiens complexes avec structures locales. Cette section examine certains des travaux dans ce domaine. Les solutions sont supposées être représentées par des chaînes binaires de longueur fixe n, bien que la plupart des méthodes présentées ici peuvent être étendues à l'optimisation de chaînes sur un alphabet fini arbitraire.

3.3.1.1 Interactions inexistantes

EGA [93] et Population-Based Incremental Learning (PBIL) remplacent la population de solutions représentées comme des chaines binaires de longueur fixe par un vecteur de probabilité $(p_1, p_2, \ldots, p_n)$, où n est le nombre de bits dans une chaîne et p_i représente la probabilité d'un '1' dans la i-ème position de chaînes de solution. Chaque p_i est initialement fixé à $0,5$, ce qui correspond à une distribution uniforme sur l'ensemble des solutions. Dans chaque itération, PBIL génère s solutions selon le vecteur de probabilité courant où $s \geq 2$. Chaque valeur est généré indépendamment de son contexte (bits restants) et donc pas d'interactions sont considérées (voir Figure 3.2). La meilleure solution de l'ensemble généré de s solutions est ensuite utilisée pour mettre à jour les vecteurs de probabilités d'entrées utilisant $p_i = p_i + \lambda(x_i + p_i)$, où $\lambda \in (0, 1)$ est le taux d'apprentissage (par exemple,0.01), et x_i est le i-ème bit de la meilleure solution. Avec l'utilisation de la règle de mise à jour ci-dessus, la probabilité p_i d'un 1 dans la i-ème position augmente si la meilleure solution contient un 1 dans cette position et diminue autrement. En d'autres termes, les vecteurs de probabilités d'entrées se déplacent vers la meilleure solution et, par conséquent, la probabilité de générer cette solution augmente. Le processus de génération de nouvelles solutions et de mettre à jour le vecteur de probabilité est répété jusqu'à ce qu'un des critères d'arrêt soient satisfaits ; par exemple, l'exécution peut être terminée si toutes les entrées de vecteur de probabilité sont suffisamment proches de 0 ou 1.

PBIL est un EDA incrémental, car il procède par l'exécution de mises à jour incrémentielles du modèle à l'aide d'un petit échantillon de solutions. Cependant, il y a une forte corrélation entre le taux d'apprentissage dans PBIL et la taille de la population dans les EDA basés population ou d'autres algorithmes évolutionnaires ; en diminuant le taux d'apprentissage λ correspond à l'augmentation de la taille de la population.

Le cGA [60] réduit l'écart entre PBIL et les algorithmes génétiques traditionnelles. Comme PBIL, cGA remplace la population par un vecteur de probabilité et toutes les entrées dans le vecteur de probabilité sont initialisées à $0,5$. Chaque itération met à jour le vecteur de probabilité en imitant l'effet d'une seule compétition entre deux solutions échantillonnées où le meilleur remplace le pire dans une population hypothétique de

taille N. En notant le bit dans la position i de la meilleure et du pire des deux solutions échantillonnées x_i et y_i, respectivement, les entrées probabilités vecteurs sont mis à jour comme suit :

$$p_i = \begin{cases} p_i + \frac{1}{N} & si\ x_i = 1 \quad et \quad y_i = 0 \\ p_i - \frac{1}{N} & si\ x_i = 1 \quad et \quad y_i = 1 \\ p_i & sinon \end{cases} \tag{3.6}$$

Bien que cGA utilise le vecteur de probabilité à la place d'une population, les mises à jour du vecteur de probabilité correspondent à remplacer une solution par une autre en utilisant une population de taille N et en réarrangeant la population résultant en utilisant un modèle univarié qui assume l'entière indépendance des variables du problème.

L'algorithme UMDA [99] maintient une population de solutions. Chaque itération de UMDA commence par la sélection d'une population de solutions prometteuses en utilisant une méthode de sélection arbitraire d'algorithmes évolutionnaires. Un vecteur de probabilité est ensuite calculé en utilisant la population sélectionnée de solutions prometteuses et de nouvelles solutions sont générées par échantillonnage du vecteur de probabilité.

Les nouvelles solutions remplacent les anciennes et le processus est répété jusqu'à ce que un critère d'arrêt est satisfait. Bien que, UMDA utilise un modèle probabiliste comme une étape intermédiaire entre les populations d'origine et les nouveaux contrairement PBIL et cGA, la performance, la dynamique et les limites de PBIL, cGA, et UMDA sont similaires.

3.3.1.2 Interactions en paire

Les EDAs basés sur des modèles probabilistes en paires, comme une chaîne, un arbre ou une forêt, représentent la première étape vers des EDA capable d'apprendre les interactions entre variables et donc résoudre les problèmes décomposables d'une manière évolutive.

L'algorithme MIMIC [34] utilise une distribution de chaîne (Figure 3.2) spécifié par

1. un ordre de positions de chaîne (variables),
2. une probabilité de 1 dans la première position de la chaîne
3. les probabilités conditionnelles de chaque autre position donnée de la valeur dans la position précédente de la chaîne.

Un modèle de chaîne probabiliste code la distribution de probabilité où toutes les po-

sitions à l'exception de la première condition dépendent de la position précédente dans la chaîne. Après avoir sélectionnée les solutions prometteuses et calculer les probabilités marginales et conditionnelles, MIMIC utilise un algorithme glouton pour maximiser l'information mutuelle entre les positions adjacentes dans la chaîne. Néanmoins, l'algorithme glouton ne garantit pas l'optimalité globale du modèle construit. L'algorithme glouton commence dans la position avec l'entropie inconditionnelle minimum. La chaîne est élargie par l'ajout d'une nouvelle position qui minimise l'entropie conditionnelle de la nouvelle variable. Une fois la chaîne complète est construite pour la population sélectionnée de solutions prometteuses, de nouvelles solutions sont générées par l'échantillonnage de distribution codée par la chaîne. L'utilisation d'interactions en paire était l'une des étapes les plus importantes dans le développement des EDAs. MIMIC a été le premier EDA discrète pour non seulement apprendre et d'utiliser un ensemble fixe de statistiques, mais il était aussi capable d'identifier les statistiques qui devraient être considérées pour résoudre le problème de manière efficace.

Baluja et Davies [64] utilisent les arbres de dépendance (Figure 3.2) pour modéliser des solutions prometteuses. Comme dans PBIL, la population est remplacée par un vecteur de probabilité mais dans ce cas le vecteur de probabilité contient toutes les probabilités de paires. Les probabilités sont initialisées à $0,25$. Chaque itération ajuste le vecteur de probabilité selon de nouvelles solutions prometteuses acquises. Un arbre de dépendance encode la distribution de probabilité où chaque variable, sauf pour la racine, est conditionnée par le parent de la variable dans l'arbre.

Les modèles en paire capturent certaines interactions dans un problème avec un temps de calcul raisonnable. EDA avec des modèles probabilistes en paires peuvent identifier, de propager et de juxtaposer des solutions partielles d'ordre 2, et donc ils peuvent bien travailler sur des problèmes décomposables en au plus deux. Néanmoins, capturer certaines interactions par paire a encore été démontré d'être insuffisante pour résoudre tous les problèmes décomposables [109].

3.3.1.3 Interactions multivariés

En utilisant des modèles multivariés permet EDA d'être capables de résoudre les problèmes difficiles rapidement, avec précision et de manière fiable [82, 105]. D'autre part, apprendre des distributions avec des interactions multivariées nécessite des algorithmes de modèles d'apprentissage plus complexes et nécessitent un temps de calcul important et ne garantissent toujours pas optimalité globale du modèle résultant. Néanmoins, de nombreux problèmes difficiles sont intraitables en utilisant des modèles simples et l'uti-

lisation de modèles et d'algorithmes complexes est nécessaire.

L'algorithme BOA [108] construit un réseau bayésien pour la population de solutions prometteuses (Figure 3.2) et échantillonne le réseau construit pour générer de nouvelles solutions. Dans toutes les variantes de BOA, le modèle est construit par un algorithme glouton qui ajoute de manière itérative une nouvelle dépendance dans le modèle qui maximise la qualité du modèle. D'autres opérateurs de graphes élémentaires peuvent être incorporés, mais des ajouts de pointe sont les plus importants. La construction est terminée lorsque plus aucune amélioration est possible. L'algorithme glouton utilisé pour apprendre un modèle dans BOA est similaire à celui utilisé dans ECGA. Cependant, les réseaux bayésiens peuvent coder de dépendances plus complexes que les modèles utilisés dans ECGA. Par conséquent, BOA est également applicable à des problèmes avec de dépendances qui se chevauchent. BOA apprend à la fois la structure et les probabilités du modèle. Bien que BOA ne nécessite pas de connaissances spécifiques sur les problèmes à l'avance, une information préalable sur le problème peut être incorporé en utilisant la statistique bayésienne, et l'influence relative de l'information préalable et la population de solutions prometteuses peut être réglé par l'utilisateur. [63, 121].

L'algorithme de BOA hiérarchique (hBOA) [107] étend BOA en employant les structures locales pour représenter les distributions locales au lieu d'utiliser des tableaux standards de probabilités conditionnelles. Cela permet hBOA pour représenter plus efficacement les distributions avec des interactions d'ordre élevé. En outre, hBOA intègre une technique de nichage appelé sélection par tournoi restreint [61] pour assurer la préservation de la diversité. Les deux extensions permettent hBOA de résoudre les problèmes décomposables en sous d'ordre borné sur un certain nombre de niveaux de difficulté d'une hiérarchie.

Les réseaux de Markov sont encore une autre classe de modèles qui peuvent être utilisés pour identifier et utiliser les interactions multivariées dans les EDAs. Les réseaux de Markov sont des modèles graphiques non orientés (Figure 3.2). Comparé aux réseaux bayésiens, les réseaux de Markov peuvent parfois couvrir la même distribution en utilisant moins d'arêtes dans le modèle de dépendance, mais l'échantillonnage de ces modèles devient plus compliqué que l'échantillonnage des réseaux bayésiens. Les réseaux de Markov sont utilisés, par exemple, dans le réseau de Markov EDA (MN-EDA) [119] et algorithme à estimation de la densité en utilisant des champs de Markov (DEUM) [123].

Les EDAs qui utilisent des modèles capables de couvrir des interactions multivariées peuvent résoudre un grand nombre de problèmes d'une manière extensible ; des résultats prometteurs ont été signalés sur un grand nombre de problèmes, y compris le partition-

nement de graphe [121], l'optimisation du réseau de télécommunication, l'optimisation de la grappe de silicium , la planification [90], la gestion des forêts, sac-à-dos multiobjectif , et d'autres.

3.3.2 EDAs avec vecteurs de valeur réelle

Il existe deux approches de base pour étendre les EDAs discrètes, de chaine de longueur fixe à d'autres domaines tels que les vecteurs à valeur réelle :

— Mapper l'autre représentation au domaine de chaines discrètes à longueur fixe, résoudre le problème discret, ensuite revenir à la représentation originale du problème.
— Étendre ou modifier la classe de modèles probabilistes à d'autres domaines.

Un certain nombre d'études ont été publiées au sujet de mapping des représentations à valeurs réelles en une représentation discrète dans les algorithmes évolutionnaires (Evolutionary computation) [3, 5, 6, 28, 54, 110] ; cette section se concentre sur les EDAs de la deuxième catégorie.

3.3.2.1 Distribution normale à pic unique

Le Hill Climbing stochastique avec apprentissage par des vecteurs de distributions normales (SHCLVND) [38] est une extension directe de PBIL à des vecteurs de variables à valeurs réelles en utilisant une distribution normale pour modéliser chaque variable. SHCLVND remplace la population de solutions à valeurs réelles par un vecteur de moyens $\mu = (\mu_1, \ldots, \mu_n)$ où μ_i représente une moyenne de la distribution de la i-ème variable. Le même écart-type σ est utilisé pour toutes les variables. Dans chaque génération (itération), un ensemble aléatoire de solutions est d'abord généré selon μ et σ. La meilleure solution de ce sous-ensemble est ensuite utilisé pour mettre à jour les entrées dans μ en décalant chaque μ_i vers la valeur de la i-ième variable de meilleure solution en utilisant une règle de mise à jour similaire à celle utilisée dans PBIL. En outre, chaque génération réduit l'écart type pour rendre l'exploration future d'espace de recherche plus restreint . Un algorithme similaire a été développé indépendamment par Sebag et Ducoulombier [122], qui a également discuté de plusieurs approches à l'évolution d'un écart-type pour chaque variable.

3.3.2.2 Mixtures de distribution normale

La densité de probabilité d'une distribution normale est centré autour de sa moyenne et diminue de façon exponentielle avec la distance carrée de la moyenne. S'il y a plusieurs nuages de valeurs, une distribution normale doit soit se concentrer sur un seul de ces nuages, ou il peut accepter plusieurs nuages au détriment d'inclure la région à faible densité entre eux. Dans les deux cas, la distribution résultante ne peut pas modéliser les données avec précision. Une façon d'étendre les modèles de distributions normales à pic unique standard pour permettre la couverture de plusieurs groupes de points semblables est d'utiliser une mixture de distributions normales. Chaque composant de mixture de distributions normales est une distribution normale par lui-même. Un coefficient est spécifié pour chaque composant de mixture pour désigner la probabilité qu'un point aléatoire appartient à ce composant. La fonction de densité de probabilité d'un mixture est donc calculée en multipliant la fonction de densité de chaque composante de mixture par la probabilité qu'un point aléatoire est classé dans le composant, et en ajoute ces densités pondérées ensemble.

Dans le framework IDEA, Bosman et Thierens [18] ont proposé d'utiliser de distribution conjointe des noyaux normaux, où une seule distribution normale est placée autour de chaque solution choisie. Une distribution conjointe des noyaux normaux peut donc être considérée comme une utilisation extrême de mixture de distributions avec un composant de mixture par point dans l'échantillon d'entraînement. La variance de chaque distribution normale peut être fixée à une valeur relativement faible, mais il devrait être préférable d'adapter les variances selon l'état actuel de la recherche. En utilisant les distributions de noyau correspond à l'utilisation d'une mutation à moyenne fixe nulle normalement distribuées pour chaque solution prometteuse comme on le fait souvent dans les stratégies d'évolution. C'est pourquoi c'est possible de prendre directement en place des stratégies d'adaptation de la variance de chaque noyau à partir de stratégies d'évolution.

L'utilisation de distributions normales n'est pas la seule approche pour la modélisation de distributions de valeurs réelles. Autres fonctions de densité sont fréquemment utilisées pour modéliser les distributions de probabilité de valeurs réelles, y compris les distributions de l'histogramme, les distributions d'intervalle, et d'autres.

3.3.3 EDAs pour les problèmes de permutation

Dans de nombreux problèmes, les solutions sont plus naturellement représentées par des permutations. Ceci est le cas, par exemple, dans de nombreux problèmes de planification. Ces types de problèmes contiennent souvent deux types spécifiques de caractéristiques ou contraintes que EDAs ont besoin de les capturer. La première est la position absolue d'un symbole dans une chaîne et la seconde est l'ordre relatif des symboles spécifiques. Dans certains problèmes, tels que le problème du voyageur de commerce, les contraintes relatives de commande importent le plus. Dans d'autres cas, tels que le QAP (Quadratic Assignment Problem), à la fois l'ordre relatif et les positions absolues sont importantes.

Pour résoudre des problèmes où les solutions sont des permutations d'une chaîne, Bengoetxea et al. [11] à commencer par un modèle de réseau bayésien construit en utilisant la même approche que dans EBNA. Cependant, la méthode d'échantillonnage est modifiée pour assurer que seuls les permutations valides sont générées. Cette approche a démontré la capacité de résoudre le problème de Graphe Matching inexacte. De la même façon, l'algorithme de dependency-tree EDA (dtEDA) de Pelikan et al. [94] commence avec un modèle d'arbre de dépendances et modifie l'échantillonnage afin de garantir que seuls les permutations valides sont générées. dtEDA pour des problèmes de permutation a été utilisée pour résoudre le QAPs structurés avec un grand succès. Les réseaux bayésiens et les modèles d'arbres sont capables de coder à la fois la position absolue et les contraintes relatives d'ordonnancement, bien que pour certains types de problèmes, ces modèles peuvent se révéler plutôt inefficace.

L'algorithme Edge-Histogram-Based Sampling Algorithm (EHBSA) [136] fonctionne en créant une matrice d'histogramme d'arête (Edge Histogram Matrix, EHM). Pour chaque paire de symboles, EHM stocke les probabilités que l'un de ces symboles suivra l'autre dans une permutation. Pour générer de nouvelles solutions, EHBSA commence par un symbole choisi au hasard. EHM est ensuite échantillonné à plusieurs reprises pour générer de nouveaux symboles dans la solution en normalisant les probabilités basées sur les valeurs qui ont déjà été générés. EHM ne tient pas compte aux positions ; afin d'aborder les problèmes dans lesquels les positions sont importants, EHBSA a été étendu à utiliser des modèles [115, 125, 128, 133]. Pour générer de nouvelles solutions, d'abord une chaîne aléatoire de la population a été choisi comme un modèle. De nouvelles solutions ont été ensuite générées en retirant des éléments aléatoires de la chaîne modèle et générer les parties manquantes par échantillonnage de EHM. L'algorithme résultant a été montré d'être mieux que la plupart des autres EDA sur le problème du voyageur

de commerce. Dans une autre étude, l'algorithme Node-Histogram Based Sampling Algorithm (NHBSA) a été proposé par Tsutsui et al. [134], qui a utilisé un modèle capable de mémoriser des fréquences de nœuds dans chaque position (codant pour de ce fait les contraintes de position) et également a utilisé un modèle.

Zhang et al. [114, 147, 148] ont proposé d'utiliser une mutation guidée pour optimiser les problèmes de permutation ainsi que les problèmes de graphes [148] à la fois. Dans la mutation guidée, les parties de la solution qui doivent être modifiées en utilisant un opérateur de voisinage stochastique sont identifiés par l'analyse d'un modèle probabiliste de la population de solutions prometteuses.

Conclusion

EDAs sont une classe d'algorithmes d'optimisation stochastiques qui ont gagné en popularité en raison de leur capacité à résoudre un large éventail de problèmes complexes avec une excellente performance et de scalabilité. En outre, alors que beaucoup de ces algorithmes ont été montré qu'ils accomplissent ces tâche bien avec peu d'informations ou même pas pour des problèmes spécifiques.

EDA ont leurs racines dans les domaines des algorithmes évolutionnaires et l'apprentissage automatique. Pour les algorithmes évolutionnaires, EDA utilise une population de solutions qui évolue par itérations de la sélection et de la variation. Pour l'apprentissage automatique, utilise l'idée de modèles d'apprentissage à partir de données, et il utilise les modèles qui on résulte pour guider la recherche vers de meilleures solutions. Cette approche est particulièrement puissante car elle permet à l'algorithme de recherche d'adapter au problème à résoudre, en donnant au EDAs la possibilité d'être un algorithme de recherche black-box. Comme la plupart des problèmes du monde réel ont une sorte de structure inhérente (plutôt que d'être complètement aléatoire), il ya un espoir que EDA peut apprendre une telle structure, ou au moins des parties de celui-ci, et de mettre ces connaissances à profit dans la recherche de l'optimum.

Une autre caractéristique clé des EDAs, et qui les distingue des autres métaheuristiques, réside dans le fait que la séquence de modèles probabilistes appris le long d'une exécution particulière (ou une séquence ou pistes) donne des informations importantes qui peuvent être exploitées à d'autres fins.

Ce chapitre a donné une introduction et a examiné à l'état de l'art dans la recherche des EDAs. Les concepts de base de ces algorithmes ont été présentés et une taxonomie a été décrite à partir des vues basées sur la décomposition du modèle et le type de dis-

tributions locales. Les EDAs les plus populaires proposées dans la littérature ont ensuite été interrogés selon les représentations les plus courantes pour les solutions candidates.

Chapitre 4

Algorithme à estimation de distribution basé Copule (CEDA)

Introduction

On a examiné dans les chapitres précédents les différents concepts liés à l'optimisation multiobjectif et les méthodes de résolutions associées. Les méthodes de résolution des problèmes multiobjectifs se divisent en deux classes ; les méthodes exactes et les méthodes approchées. Les méthodes exactes soufrent de plusieurs inconvénients comme exemple : la difficulté d'avoir des bonnes solutions d'un problèmes non-convexe. Alors que, les méthodes basées sur les métaheuristiques surpassent ces incapacités. Un examen des différentes méthodes montre que les algorithmes évolutionnaires sont les métaheuristiques les appropriés à résoudre cette classe de problème. Dans ce chapitre nous décrivons les différentes étapes de notre proposition qui est un algorithme évolutionnaire multiobjectif basé sur la dominance (Voir 2.1.2). L'algorithme proposé utilise une représentation explicite des solutions qui est utilisée par une classe des EAs appelée EDA (Estimation of Distribution Algorithm, Voir 3.1). Les algorithmes à estimation de distribution utilisent des modèles pour estimer la distribution de meilleures solutions trouvées, ces modèles vont être utilisés pour reproduire des nouvelles solutions. Dans le chapitre 3 on a présenté les différentes méthodes de modélisation de dépendances et les représentations possibles. Parmi les représentations utilisées dans les problèmes de monde réel sont les représentations à vecteur de valeurs réelles.

Récemment, une nouvelle approche pour développer les EDAs pour résoudre le problème d'optimisation à valeur réelle a été développée qui est basée sur la théorie de la

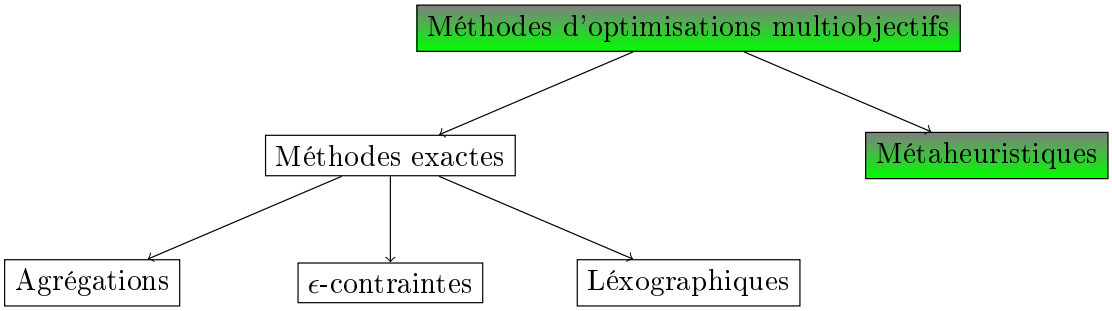


FIGURE 4.1: Classification des méthodes de résolutions d'un problème multiobjectif.

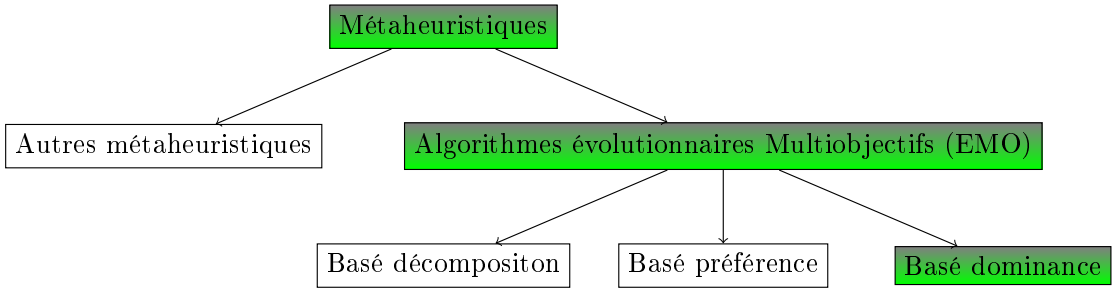


FIGURE 4.2: Métaheuristiques pour l'optimisation multiobjectif.

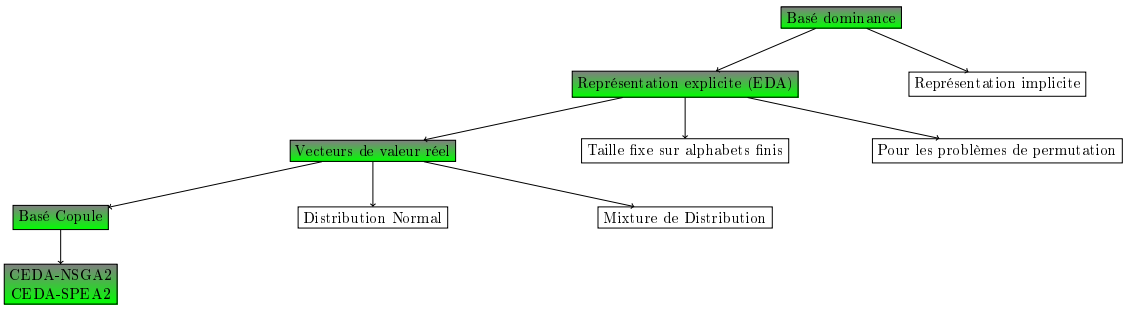


FIGURE 4.3: Méthodes de représentation et reproduction dans les algorithmes évolutionnaires.

copule. L'idée principale de Copules est de décomposer la distribution conjointe multivariée dans chaque distribution univariée et une copule. La copule est une fonction qui capture la dépendance entre les variables. L'utilisation de modèles basés-copule dans les EDAs continues place ces algorithmes dans une position avantageuse par rapport à d'autres EDAs qui reposent sur les distributions multivariées, comme la distribution normale multivariée 3.3.2. Par le biais de Copules, toute distribution multivariée peut être décomposé en une distribution marginale et la copule qui détermine la structure de dépendance entre les variables.

La Figure 4.1,4.2 et 4.3 montrent les liens entre les différents choix des méthodes de résolution jusqu'à CEDA-(NSGA-II et SPEA2). Dans la suite de ce chapitre, nous expliquons les principales notions liées aux copules. Ensuite, nous présentons en détail notre proposition CEDA avec les deux méthodes de sélection NSGA-II et SPEA2.

4.1 Les Mesures de dépendances

Dans cette section, les propriétés souhaitables des mesures de dépendance sont discutés. En outre, différents types de mesures de dépendance seront abordés à savoir : la corrélation linéaire de Pearson et la corrélation de rang. Depuis notre travail de recherche se concentre sur les copules bivariées, la discussion de la dépendance sera limitée au cas bivariées.

4.1.1 Les propriétés souhaitables des mesures de dépendance

Avant de discuter les propriétés souhaitables de mesure de dépendance, une introduction à la notion de dépendance est nécessaire. Deux variables aléatoires X et Y sont dits dépendantes ou associées si elles ne satisfont pas la propriété d'indépendance : $(X, Y) = F_1(X) * F_2(Y)$, où $F_1(X)$ et $F_2(Y)$ sont les distributions marginales des variables aléatoires X et Y. Soit la valeur δ qui exprime une simple mesure de dépendance scalaire . Quatre propriétés souhaitées de δ sont décrits par [46] .

1. $\delta(X, Y) = \delta(Y, X)$, connue comme la condition de symétrie.
2. $-1 \leq \delta(X, Y) \leq 1$, connue comme la condition de normalisation.
3. $\delta(X, Y) = \begin{cases} 1 & \text{alors} \quad (X, Y) \text{ sont comonotone.} \\ -1 & \text{alors} \quad (X, Y) \text{ sont anti-comonotone.} \end{cases}$
4. $\delta(T(X), Y) = \begin{cases} \delta(X, Y) & T \text{ croissante} \\ -\delta(X, Y) & T \text{ décroissante} \end{cases}$

Pour une transformation strictement monotone $T : \mathbb{R} \to \mathbb{R}$ de X.

4.1.2 Corrélation linéaire de Pearson

La corrélation linéaire de Pearson est le type le plus utilisé des mesures de dépendance. La corrélation linéaire de Pearson mesure la direction et la mesure dans laquelle une

variable est en relation linéaire avec une autre variable. Le coefficient de corrélation linéaire est définie par :

$$\rho(X,Y) = \frac{Cov[X,Y]}{\sigma_X \sigma_Y} \tag{4.1}$$

où $Cov[X,Y]$ est la covariance en X et Y, σ_X σ_Y sont les écart-types des variables X et Y, respectivement. La valeur de coefficient ρ varie entre les valeurs -1 et 1. Si $\rho = 1$ alors les variables X et Y sont dits parfaitement dépendante par une relation croissante. Si $\rho = -1$ alors les variables X et Y sont tout à fait dépendante par une relation décroissante. En outre, si les variables aléatoires X et Y sont indépendants alors la corrélation entre ces deux variables est égal à zéro.

Par ailleurs, Si on considère les deux variables, $X \sim N(0,1)$ et $Y = X^2$, dans ce cas, la corrélation entre les deux variables X et Y égale à zéro alors que X et Y sont parfaitement dépendantes.

4.1.3 Corrélation des rangs

Pour remédier à la faiblesse de la corrélation linéaire de Pearson, deux mesures de corrélation des rangs seront discutées à savoir : le rang de Spearman (ρ_s)et τ de Kendall. Parce que ces deux mesures reposent sur la notion de concordance, ce concept sera présenté tout d'abord. Les observations (X_1, Y_1) et (X_2, Y_2) sont dites d'être concordante si $(X_1 < X_2)$ et $(Y_1 < Y_2)$ ou si $(X_1 > X_2)$ et $(Y_1 > Y_2)$. Cela signifie que les grandes valeurs (petits) de la variable aléatoire X sont associées à des valeurs élevées (petits) de la variable aléatoire Y. Si le contraire est vrai, une discordance se pose. Dans ce qui suit les deux mesures de corrélation des rangs seront discutées plus en détail.

4.1.3.1 Corrélation de Spearman

La corrélation de spearman linéaire est une mesure de corrélation non-paramétrique définie par [45] :

$$\rho_s = 1 - \frac{6 \sum d^2}{n(n^2-1)} \tag{4.2}$$

Où n est le nombre de paires dans les rangs, et d est la différence entre les rangs appariés. En ce sens, le rang de corrélation de Spearman peut être considéré comme la corrélation linéaire de Pearson entre les variables classées. Les variables sont classées en

assignant le rang le plus élevé à la plus haute valeur.

4.1.3.2 Tau de Kendall

Tau de Kendall mesure la différence entre la probabilité de concordance et celui de discordance entre les variables aléatoires X et Y [30].

$$\tau = \frac{C - D}{n(n-1)/2} \tag{4.3}$$

Où C est le nombre de paires concordantes et D est le nombre de paires discordantes.

4.2 Fonctions de distributions et variables aléatoires

4.2.1 Densité de probabilité PDF

La fonction de densité de probabilité (PDF) f_X d'un variable aléatoire continue X est une fonction qui décrit la probabilité de la variable aléatoire à prendre sur une certaine valeur. L'intégrale de la PDF sur une certaine plage donne la probabilité que la variable aléatoire prenne une valeur dans cet intervalle. Le PDF est notée $f_X(x)$ et représenté par :

$$P(a < X < b) = \int_a^b f_X(x)dm \tag{4.4}$$

Certaines caractéristiques de PDFs sont les suivantes :

- $f_X(x)$ est toujours positif, $f_X(x) \geq 0$, pour toutes $X \in (-\infty, \infty)$.
- La surface sous la courbe $f_X(x) \in (-\infty, \infty)$ est égale à 1.

$$P(-\infty < X < \infty) = \int_{-\infty}^{\infty} f_X(x)dx = 1 \tag{4.5}$$

La fonction de densité bivariée de X et Y a la forme générale de $f_{XY}(x, y)$. Elle est donnée par :

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_{XY}(x, y)dxdy = 1 \tag{4.6}$$

4.2.2 Fonction de répartition CDF

La fonction de répartition (Cumulatif Distribution Function, CDF) d'une variable aléatoire X est donnée par :

$$F_X(x) = P(X \leq x) \tag{4.7}$$

Le CDF est donné par l'équation :

$$F_X(x) = \int_{-\infty}^{x} f_X(x)dx \tag{4.8}$$

Les propriétés de CDF sont données par :

$$0 \leq F(x) \leq 1, \text{quand } x \in (-\infty, \infty), \tag{4.9}$$

$$F(-\infty) = 0, F(\infty) = 1, \tag{4.10}$$

$$\text{si } x_1 < x \text{ alors } F(x_2) - F(x_1), \tag{4.11}$$

$$P(x_1 < x < x_2) = F(x_2) - F(x_1) \tag{4.12}$$

La distribution de probabilité conjointe de X et Y est exprimée avec la fonction de densité (CDF) :

$$F_{XY}(x, y) = P(X \leq x, Y \leq y) = \int_{-\infty}^{y} \int_{-\infty}^{x} f_{XY}(x, y)dxdy \tag{4.13}$$

4.2.3 Loi de probabilité marginale

Le comportement stochastique des données univariées est décrit par sa distribution de probabilité. Pour le cas bivariée, la distribution marginale de X est la probabilité de X sur Y. L'intégration de la distribution de probabilité conjointe sur y donne :

$$F_X(x) = \int_y F_{XY}(x, y)dy = \int_y F_{X|Y}(x|y)F_y(y)dy \tag{4.14}$$

Où $F_{XY}(x, y)$ est la distribution conjointe de X et Y, $F_{X|Y}(x|y)$ est une distribution conditionnelle.

4.3 Théorie de Copule

Sklar [129] introduit les fonctions de Copule comme un outil puissant pour modéliser la dépendance entre les variables. Le nom "Copule" est un mot latin qui signifie un lien. Les méthodes basées copule ne nécessitent pas toutes les hypothèses concernant les distributions de probabilité sous-jacentes des données. La méthode exploite les relations linéaires et non-linéaires entre deux ou plusieurs variables en ajustant une fonction de copule sur une distribution bivariée ou multivariée empirique. Enfin, de nouvelles données qui sont compatible avec la structure de dépendance précédemment dérivée peut être simulée par l'évaluation de la fonction de répartition qui est donnée par la copule.

4.3.1 Le théorème de Sklar

Le théorème de Sklar indique qu'une copule C relie une fonction de distribution multivariée donnée avec ses marginaux univariée. Pour une distribution bivariée, pour la distribution conjointe $F_{XY}(x, y)$ avec des fonctions de distribution marginale univariée $F_X(x)$ et $F_Y(y)$, il existe une copule C.

$$F_{XY}(x, y) = C(F_X(x), F_Y(y)) = C(u, v) \tag{4.15}$$

$$C : [0, 1]^2 \to [0, 1] \tag{4.16}$$

si C est une copule et F_X et F_Y sont des fonctions de distribution, F_{XY} peut être définie comme une distribution conjointe avec F_X et F_Y. La fonction de densité (PDF) de la distribution bivariée $f_{X,Y}(x, y)$ est représentée par :

$$f_{XY}(x, y) = C(F_X(x), F_Y(y)) \cdot f_X(x) \cdot f_Y(y) \tag{4.17}$$

où $f_X(x)$ et $f_Y(y)$ sont les lois marginales univariée de X et Y. La Copule est unique lorsque les marginales sont continues.

4.3.2 Caractéristique de Copules

Quelques caractéristiques de Copules sont données par Genest et Rivest [52] et Jaworski et al. [**?**] . Dans le cas bivarié, une copule est représentée comme une fonction C

à partir de $[0,1]^2$ vers $[0,1]$ de telle sorte que $\forall u, v \in [0,1]$:

$$C(u,0) = C(0,v) = 0 \tag{4.18}$$

$$C(u,1) = u \text{ et } C(1,v) = v \tag{4.19}$$

La Copule est une fonction croissante. Elle implique que $\forall u_1, u_2, v_1, v_2 \in [0,1]$ avec $u_1 \leq u2$ et $v_1 \leq v_2$ donne

$$C(u_2,v_2) - C(u_2,v_1) - C(u_1,v_2) - C(u_1,v_1) \geq 0 \tag{4.20}$$

La Copule est une fonction continue :

$$|C(u_2,v_2) - C(u_1,v_1)| \leq |u_2 - u_1| - |v_2 - v_1| \tag{4.21}$$

4.3.3 Copule Archimedean

Une copule Archimedean peut être présentée sous la forme suivante :

$$C(u_1,\ldots,u_d;\theta) = \phi^{-1}\{\phi(u_1) + \cdots + \phi(u_d), \theta\} \tag{4.22}$$

où θ est le paramètre de copule et la fonction ϕ est le générateur de la copule avec les caractéristiques suivantes :

- pour tout $u \in (0,1), \phi(u) < 0, \phi$ est décroissante
- pour tout $u \in (0,1), \phi(u) < 0, \phi$ est convexe
- $\phi(0) = \infty$
- $\phi(1) = 0$

est ϕ^{-1} est définie par

$$\phi^{-1} = \begin{cases} \phi^{-1}(t;\theta), & \text{si} \quad 0 \leq \phi \leq \phi(0) \\ 0, & \text{si} \quad \phi(0) \leq \phi \leq \infty \end{cases} \tag{4.23}$$

pour le cas bivarié on a

$$C(u,v) = \phi^{-1}(\phi(u), \phi(v)). \tag{4.24}$$

Ils existent trois copules Archimedean couramment utilisées, ces copules seront expliquées dans la suite.

4.3.3.1 Copule de Frank

La fonction de génération de la Copule de Frank est définie par

$$\phi(t) = -\ln\left(\frac{e^{-\theta t}-1}{e^{-\theta}-1}\right) \tag{4.25}$$

Le paramètre θ est définie sur $(-\infty,\infty)-\{0\}$. La Copule de Frank est donnée alors par :

$$C_\theta(u,v) = \frac{1}{\theta}\ln\left(1+\frac{(e^{-\theta u}-1)(e^{-\theta v})}{e^{-\theta}-1}\right) \tag{4.26}$$

La Copule de Frank permet de modéliser les données avec une dépendance positive et négative. Les grandes valeurs positives et négatives de θ indiquent une forte dépendance et $\theta = 1$ implique une indépendance totale. La Copule de Frank est appropriée pour deux ensembles de données avec des mêmes caractéristiques dynamiques. La relation entre le paramètre de copule de Frank et le tau de Kendall est donné par :

$$\tau = 1+\frac{4[D_1(\theta)-1]}{\theta} \text{ avec } D_k(\alpha) = \frac{k}{\alpha^k}\int_0^\alpha \frac{t^k}{exp(t)-1}dt \tag{4.27}$$

4.3.3.2 Copule de Clayton

La fonction génératrice de la Copule de Clayton est définie par :

$$\phi(t) = \frac{1}{\theta}(u^{-\theta}-1) \tag{4.28}$$

Par conséquent, la Copule de Clayton est définie comme suit :

$$C_\theta(u,v) = (u^{-\theta}+v^{-\theta}-1)^{-\frac{1}{\theta}} \tag{4.29}$$

où θ est limité sur l'intervalle $[-1,\infty)$. Si $\theta = 0$, il montre le cas d'indépendance et quand $\theta \to \infty$, indique une forte dépendance dans l'espace de rang inférieur. La dépendance entre le paramètre θ et le tau de Kendall est simplement donnée par :

$$\tau = \frac{\theta}{\theta+2} \tag{4.30}$$

4.3.3.3 Copule de Gumbel

La Copule de Gumbel (1960) est utilisée pour modéliser la relation asymétrique dans les données [75]. Le générateur de copule Gumbel est définie par :

$$\phi(t) = (-\ln t)^{\theta} \tag{4.31}$$

La Copule de Gumbel est définie comme suit :

$$C_{\theta}(u,v) = e^{-((-\ln(u)^{\theta})+(-\ln(v)^{\theta}))^{\frac{1}{\theta}}} \tag{4.32}$$

Le paramètre de copule θ est sur l'intervalle $[1,\infty)$. Si θ est égal à 1, la Copula montre une indépendance. Lorsque $\theta \to \infty$, la copule de Gumbel indique une forte dépendance entre les variables aléatoires. La relation entre le paramètre de copule de Gumbel et le tau de Kendall est donné par :

$$\tau = 1 - \theta{-}1 \tag{4.33}$$

Un résumé des copules archimedeans et ces fonctions génératrices et paramètres sont décrites dans le tableau 4.1.

Famille	Générateur	Paramètre	Formule
Clayton	$\phi(t) = \frac{1}{\theta}(u^{-\theta} - 1)$	$-1 \leq \theta$	$C_{\theta}(u,v) = (u^{-\theta} + v^{-\theta} - 1)^{-\frac{1}{\theta}}$
Frank	$\phi(t) = -\ln\left(\frac{e^{-\theta t}-1}{e^{-\theta}-1}\right)$	$-\infty < \theta < \infty$	$C_{\theta}(u,v) = \frac{1}{\theta}\ln\left(1 + \frac{(e^{-\theta u}-1)(e^{-\theta v})}{e^{-\theta}-1}\right)$
Gumbel	$\phi(t) = (-\ln t)^{\theta}$	$1 \leq \theta$	$C_{\theta}(u,v) = e^{-((-\ln(u)^{\theta})+(-\ln(v)^{\theta}))^{\frac{1}{\theta}}}$

TABLE 4.1: Trois familles de Copule Archimedean (Clayton, Frank, et Copule de Gumbel) et leur générateur, l'espace des paramètres, et leur formule. θ est le paramètre de la Copule et aussi appelé le paramètre de dépendance, qui mesure la dépendance entre la marginal.

4.3.4 Simulation de Copules

Les Copules ont des applications directes primaires et à la simulation de variables dépendantes. Nous présentons maintenant les procédures générales pour simuler des variables dépendantes bivariées. Tout d'abord nous avons besoin d'introduire la notion de pseudo-inverse d'une fonction de distribution.

Définition 5 *Soit F une fonction de distribution univariée. Un pseudo-inverse de F est une fonction $F^{[-1]} : \mathbb{I} \to \overline{\mathbb{R}}$ tel que :*

1. *si $t \in Ran(F)$, alors $F^{[-1]}(t)$ est un nombre $x \in \overline{\mathbb{R}}$ tel que $F(x) = t$, à savoir, pour tout $t \in Ran(F)$ $F(F^{[-1]}(t)) = t$*
2. *si $t \notin Ran(F)$ alors*

$$F^{[-1]}(t) = \inf\{x : F(x) \geq t\} = \sup\{x : F(x) \leq t\} \tag{4.34}$$

Il est clair que si F est strictement croissante,alors il a une seule pseudo-inverse, qui est égale à la fonction inverse F^{-1}.

Un algorithme général pour générer des observations (x, y) d'une paire de variables aléatoires (X, Y) avec des marginaux F_X, F_Y, distribution conjoint F_{XY}, et un Copule C est comme suit. Premièrement, on génère une paire d'observation uniforme (u, v) de variables aléatoires (U, V) et ayant la Copule C. Ensuite, en utilisant la méthode de la transformée inverse, on transforme (u, v) en (x, y) à savoir :

$$\begin{cases} x = F_X^{[-1]}(u) \\ y = F_Y^{[-1]}(v) \end{cases} \tag{4.35}$$

Un pseudo-algorithme peut être vu comme suit :

— Générer des variables aléatoires indépendantes u, t uniforme sur $\mathbb{I}$.

— Définir $v = C^{[-1]}(t)$.

La paire désirée est alors (u, v).

4.4 Algorithme à estimation de distribution basé copule

Dans cette section nous expliquons les différentes étapes de l'algorithme à estimation de distribution proposé. L'algorithme utilise les outils de modélisation de dépendance déjà mentionnés dans la section précédente. L'algorithme CEDA (Copula-based Estimation of Distribution Algoritm) passe comme tout algorithme évolutionnaire par les deux étapes de sélection et la reproduction, ces étapes vont être expliqué en détail dans les paragraphes qui suit.

4.4.1 Idée de la proposition

L'algorithme CEDA (Copula-based Estimation of Distribution Algoritm), est un algorithme à estimation de distribution qui utilise la capacité des copules à modéliser les dépendances entre les solutions Pareto obtenus, ces copules vont être utilisés ensuite pour reproduire des individus dans la génération suivante.

Les copules utilisées dans notre algorithme CEDA proposé sont les copules Archimedeans. Le choix de ce type de copule réside dans le fait qu'ils modélisent la dépendance des variables par une copule avec un seul paramètre θ comme il est indiqué dans la Section 4.3.3.

De plus, en utilise une copule à deux variables (bivariate copula) pour des raisons de simplicité d'implémentation. Les variables d'entrées de la copule sont des solutions Pareto. Il faut noter que les solutions Pareto X sont les variables de décision des problèmes utilisés, qui sont dans la plupart de temps de dimension supérieure à deux. Cette propriété, nous a fait penser d'une manière d'adapter l'utilisation d'une copule bivariée sur des variables de dimension plus de deux. Pour résoudre ce problème il faut avoir pour une copule deux variables et chaque variable est de N dimension.

Dans notre travail, on a supposé qu'une dépendance entre les solutions Pareto existe, alors on a divisé l'ensemble des solutions Pareto en deux groupes d'une manière aléatoire et essayer de capturer la dépendance avec plusieurs copules Archimedean bivariées, la Section 4.4.5.1 explique la manière de créer les copules.

4.4.2 Framework général

Notre algorithme proposé est algorithme évolutionnaire (algorithme 7) alors il comporte deux étapes principales : (i) la sélection et (ii) la reproduction. Dans l'étape de sélection (interprété par Fonction SelectUsingMOEA), nous utilisons NSGA2 [36] ou SPEA2 [157] pour sélectionner les meilleurs individus (solutions) qui seront utilisés dans l'étape de reproduction où nous faisons usage de Copules pour estimer et régénérer de nouveaux individus (effectuées par les fonctions ConstructCopulas et GenerateSolutions respectivement).

Algorithme 7 Copula-based EDA

function CEDA
  $\mathbf{P_0} \leftarrow$ Initialization(m)
  $\mathbf{P} \leftarrow$ SelectUsingMOEA($\mathbf{P_0}$) ▷ MOEA peut être NSGA2 ou SPEA2
  tant que critère d'arrêt **faire**
    C $\leftarrow$ ConstructCopulas($\mathbf{P}$)
    $\mathbf{P'} \leftarrow$ GenerateSolutions(C)
    $\mathbf{P''} \leftarrow$ SelectUsingMOEA($[\mathbf{P'P}]^T$) ▷ $[\mathbf{P'P}]^T$ prendre tous les individus de $\mathbf{P'}$ et $\mathbf{P}$
    $\mathbf{P} \leftarrow \mathbf{P''}$
  fin tant que
  return $(\mathbf{P}, C)$
fin function

Un pseudo-code de notre algorithme qui effectue l'estimation de la distribution en utilisant une copule pour résoudre les problèmes multiobjectifs peut être considéré comme suit (algorithme 7).

4.4.3 Initialisation

Au départ, nous supposons que nous avons une population $\mathbf{P_0} = [\mathbf{x_1}, \ldots, \mathbf{x_m}]^T$ où $\mathbf{x_i}, i \in [1, m]$ sont les individus. Chaque individu $\mathbf{x_i} = [x_{i1}, \ldots, x_{in}]$ où $x_{min} \leq x_{ij} \leq x_{max}$. x_{min} et x_{max} sont des valeurs réelles. Où chaque x_{ij} est initialement produit selon une répartition uniforme dans $[x_{min}, x_{max}]$. Notez que chaque individu $\mathbf{x_i}, (i \in [1, m])$ est un vecteur de valeur réelle, à savoir tous les $x_{ij}, (i \in [1, m], j \in [1, n])$ sont des valeurs réelles.

4.4.4 Sélection

Dans l'étape de sélection réalisée par la fonction `SelectUsing`MOEA, nous utilisons l'un des algorithmes NSGA2 ou SPEA2 classique comme un MOEA.

Le résultat de la sélection est un ensemble d'individus qui seront utilisés dans l'étape de reproduction. Nous appelons $\mathbf{P}$ la matrice des individus résultant du processus de sélection appliqué sur la population précédente. Pour la première génération, nous utilisons la population initiale $\mathbf{P_0}$. Nous avons défini $\mathbf{P}$ comme suit :

$$\mathbf{P} = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix} \tag{4.36}$$

NSGA2 ou SPEA2 sélectionne les meilleurs individus de la population, il fonctionne selon la relation de dominance défini dans la Section 1.1. Notez que les solutions générées (obtenu par `GenerateSolutions`) à une étape donnée ne sont pas nécessairement meilleure que celles générées lors de l'étape qui l'a précédée. Par conséquent, la sélection des meilleures solutions fonctionne sur l'union des deux ensembles : les solutions obtenues à partir de l'étape en cours et ceux résultats de l'étape qui l'a précédée, comme indiquée dans l'algorithme 7.

4.4.5 Reproduction

Pour effectuer la reproduction, nous commençons par le calcul de la dépendance entre les meilleurs individus en utilisant les copules comme indiqué dans l'algorithme 8, puis générer de nouvelles individus en utilisant ces Copules comme indiqué dans l'algorithme 9.

4.4.5.1 Construction des Copules

Le type de copule utilisée dans le présent document est la copule Archimedean. Les copules Archimedean traitent de deux vecteurs de variables $\mathbf{u}$ et $\mathbf{v}$; Par conséquent, nous avons divisé chacun de nos vecteurs de variables de décision en deux sous vecteurs pour tenir dans les variables $\mathbf{u}$ et $\mathbf{v}$. Nous avons effectué cette division en deux sous vecteurs à chaque génération de l'algorithme.

Nous opérons sur la transposé de la matrice $\mathbf{P}$, c.-à-d., $\mathbf{P}^T$. Nous prenons chaque vecteur $\mathbf{w_j}(j \in [1, m])$ depuis $\mathbf{P}^T$ pour construire nos sous vecteurs selon la formule suivante : $\mathbf{u_1}, \mathbf{v_1}$ sont extraites de $\mathbf{w_1}$ la taille de chacun des $\mathbf{u_1}$ et $\mathbf{v_1}$ sont égaux à $m/2$. Les éléments de $\mathbf{u_1}$ sont prélevés au hasard à partir de $\mathbf{w_1}$, et $\mathbf{v_1}$ est construit à partir du reste des éléments de $\mathbf{w_1}$. Par souci de simplicité, nous supposons que m est un nombre pair. Dans le cas où m est impair, on enlève un individu de la population initiale pour rendre sa taille paire. Le calcul des autres sous vecteurs $\mathbf{u_2}, \ldots, \mathbf{u_n}$ et $\mathbf{v_2}, \ldots, \mathbf{v_n}$ est effectuée d'une manière similaire à celle de $\mathbf{u_1}$ et $\mathbf{v_1}$ respectivement.

Nous créons les Copules Archimedean représentées par le vecteur $C = [C_1 \dots C_j \dots C_n]$, en utilisant les sous vecteurs $\mathbf{u_1}, \dots, \mathbf{u_n}, \mathbf{v_1}, \dots, \mathbf{v_n}$, où chaque copule $C_j, j \in [1, n]$ est construit à partir du sous vecteurs $\mathbf{u_j}$ et $\mathbf{v_j}$ comme indiqué dans l'algorithme 8.

Algorithme 8 Construct Copulas

function ConstructCopulas($\mathbf{P}, type$) ▷ $type$ soit :Frank, Clayton ou Gumbel
  pour tout $\mathbf{w_j}$ **un vecteur dans** $\mathbf{P^T}$ **faire**
    $\mathbf{u_j} \leftarrow$ RandomPick($\mathbf{w_j}$) ▷ Choix aléatoire de demi éléments de $\mathbf{w_j}$
    $\mathbf{v_j} \leftarrow$ Remainder($\mathbf{w_j}, \mathbf{u_j}$) ▷ Prendre le reste de $\mathbf{w_j}$
    $\mathbf{C_j} \leftarrow$ Copula($\mathbf{u_j}, \mathbf{v_j}, type$)
  fin pour
  return $C = [C_1 \dots C_j \dots C_n]$
fin function

Notez qu'il existe de nombreux types de Copules Archimedean. Dans ce travail, nous avons examiné trois d'entre eux à savoir Gumbel, Frank et Clayton. Nous avons expérimenté avec les trois types sur divers problèmes d'optimisation et on a constaté que la Copules Frank donne de meilleurs résultats.

4.4.5.2 Génération de nouveaux individus

Nous utilisons les Copules crées $C_1, \dots, C_n$ pour générer de nouveaux individus. L'ensemble des nouveaux individus générés $\mathbf{X'} = [\mathbf{w'_1}, \dots, \mathbf{w'_n}]$ où $\mathbf{w'_j}$ est la concaténation de $\mathbf{u'_j}$ et $\mathbf{v'_j}$ qui sont échantillonnés en utilisant la copule C_j. Notez que le vecteur $\mathbf{w'_j}$ (résultant de la concaténation de $\mathbf{u'_j}$ and $\mathbf{v'_j}$) est d'une taille m' qui ne soit pas nécessairement le même de la taille m de la population initiale. Les nouveaux individus sont donc les vecteurs $\mathbf{x'_i}, i \in [1, m']$ où $\mathbf{X'} = [\mathbf{x'_1}, \dots, \mathbf{x'_{m'}}]^T$. L'algorithme 9 résume ces étapes.

Algorithme 9 Generate Solutions

function GenerateSolutions(C, m') ▷ m' nombre des individus à générer
  pour tout C_j **dans** C **faire**
    $(\mathbf{u'_j}, \mathbf{v'_j}) \leftarrow$ GenerateFromCopula(C_j, m') ▷ $\mathbf{u'_j}$ et $\mathbf{v'_j}$ de taille $m'/2$.
    $\mathbf{w'_j} \leftarrow$ Concat($\mathbf{u'_j}, \mathbf{v'_j}$)
  fin pour
  return $\mathbf{X'} = [\mathbf{w'_1} \dots \mathbf{w'_j} \dots \mathbf{w'_n}]$
fin function

4.5 Diversification a posteriori des Solutions

Dans cette section nous expliquons notre contribution dans le cas où un décideur n'est pas satisfait des solutions obtenues lors de la phase d'optimisation. L'objectif de cette proposition est d'aider le décideur d'obtenir les solutions qui lui convient. Pour atteindre cet objectif, nous proposons un algorithme en deux étapes qui est composé de l'étape d'optimisation qui trouve un ensemble des meilleures solutions à un problème donné (voir section 4.4.2) et la phase de mise à jour qui trouve un autre ensemble des meilleures solutions jusqu'à le décideur est satisfait (voir section 5.3.2). A noter que l'étape de mise à jour s'exécute beaucoup plus rapidement pour trouver de nouvelles solutions par rapport à la phase d'optimisation initiale.

La phase d'optimisation nous a permis de calculer de nouvelles solutions, comme indiqué dans l'algorithme 7. Ces solutions peuvent ne pas répondre aux besoins du décideur et donc une autre étape de la génération de nouvelles solutions est nécessaire. L'étape de mise à jour que nous proposons dans notre travail (comme indiqué dans l'algorithme 10) rend possible pour le décideur de trouver d'autres nouvelles solutions rapidement en utilisant les Copules construites dans la phase d'optimisation. Plus précisément, nous atteignons cet objectif en appelant la fonction `GenerateSolutions` avec les arguments C (les copules construites dans la phase d'optimisation), et m'' est le nombre de nouveaux individus requis. La sortie de la phase de mise à jour est $\mathbf{P}_{\text{update}}$. Si le décideur n'est pas toujours satisfait des solutions obtenues, seul un autre tour de la phase de mise à jour est nécessaire sauvant ainsi la nécessité pour l'exécution de la phase d'optimisation autre fois.

Algorithme 10 Update Solutions

1: **function** UpdateSolutions(C, m'')
2: $\quad\mathbf{P}_{\text{tmp}} \leftarrow$ GenerateSolutions(C, m'')
3: $\quad\mathbf{P}_{\text{update}} \leftarrow$ SelectUsingMOEA($\mathbf{P}_{\text{tmp}}$)
4: $\quad$**return** $\mathbf{P}_{\text{update}}$
5: **fin function**

Il est important de noter que les copules utilisée comme entrée dans l'algorithme de mise à jour de la solution ont été construits en utilisant un ensemble des meilleures solutions obtenues lors de la dernière génération de l'algorithme utilisé dans l'étape d'optimisation. Par conséquent, ces Copules caractérisent fondamentalement la répartition des meilleures solutions rendant ainsi les nouveaux individus $\mathbf{P}_{\text{tmp}}$ parmi les meilleures solutions. Les solutions de retour à ce stade (Update étape) $\mathbf{P}_{\text{update}}$ sont choisis parmi

les individus temporaires $\mathbf{P}_{\text{tmp}}$ selon l'une des MOEA pour sélectionner les meilleures solutions, comme indiqué dans l'algorithme 10.

Conclusion

On a présenté dans ce chapitre trois principales contributions : 1) la proposition d'un nouveau algorithme à estimation de distribution. L'algorithme proposé utilise une méthode d'estimation très utile dans les statistiques qui est la copule et un type très célèbre qui est l' Archimedean et 2) l'application du nouveau algorithme EDA basé Copule pour résoudre des problèmes d'optimisation multiobjectif puis 3) l'utilisation du modèle obtenus pour générer des nouvelles solutions Pareto de manière très efficace.

L'algorithme EDA basé Copule utilise dans la phase d'estimation la copule Archimedean. Cette copule est le modèle utilisé pour décrire la distribution et les dépendances entre les solutions Pareto obtenues dans une génération de l'algorithme. La copule est créée en considérant que l'ensemble Pareto a une sorte de dépendance entre ces éléments. Les solutions Pareto sont utilisées comme une entrée pour la copule. Les nouveaux individus seront générés à l'aide du modèle de copule, ces individus sont évalués et les meilleurs sont sélectionnés à l'aide de l'algorithme SPEA2 ou NSGA-II. L'algorithme proposé montre de bons résultats avec les problèmes multiobjectif de référence utilisés. Nous avons fait une comparaison entre le SPEA2 classique et algorithmes NSGA-II et celle qui est proposée.

Cette comparaison a été basée sur un couple de paramètres qui sont l'hypervolume et la métrique de diversité Δ. Un nouveau aspect a été traité par l'algorithme proposé, cet aspect était la mise à jour des solutions Pareto dans un délai d'exécution très négligeable, cet aspect peut aider le DM d'obtenir de nouveaux choix, tous les nouveaux choix sont des meilleurs choix car ils sont générés à partir du dernier modèle (copules), qui est le modèle qui décrit les solutions Pareto optimales. Cet aspect est nouveau et original, car toutes les méthodes a posteriori ne peuvent pas donner au DM des nouvelles solutions Pareto sans exécuter de nouveau leur algorithme.

Chapitre 5

Algorithme Hybride pour la résolution des problèmes many-objectives

Introduction

Les problèmes d'optimisation multiobjectifs comportant plus de trois objectifs sont appelés problèmes d'optimisation "Many-objective". L'optimisation Many-objective apporte avec elle un certain nombre de défis qui doivent être abordés, ce qui souligne la nécessité de nouveaux et de meilleurs algorithmes qui peuvent gérer efficacement le nombre croissant d'objectifs. Dans ce chapitre en va présenter les différents concepts les difficultés liées à ce type de problèmes. En outre, on va citer et expliquer quelques méthodes de résolution des problèmes Many-objectives.

Par la suite, nous donnons les détails de notre proposition qui sert à trouver des nouvelles solutions d'un problème Many-objective lorsqu'un DM n'est pas satisfait par les solutions obtenues après la phase d'optimisation. La méthode utilise une hybridation entre un algorithme à estimation de distribution basé copule et un algorithme d'apprentissage automatique SVM. Cette hybridation a pour but de minimiser le temps de calcul produit par l'évaluation d'un nombre important des fonctions objectifs. Des résultats de simulation sont présentés dans ce chapitre et le reste est présenté dans l'annexe A.

5.1 Problème Many-objective

5.1.1 Définition

Les problèmes Many-objectives sont des problèmes multiobjectifs avec un nombre d'objectifs supérieurs à quatre. Alors, la définition reste la même pour un problème Many-objective [26]. Un simple problème multiobjectif est formulé comme suit :

$$\min F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x}))^T \; \mathbf{x} \in \mathbf{X} \subset \mathbb{R}^n \tag{5.1}$$

où $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ est un vecteur de n variables de décision et $\mathbf{X}$ est un espace de décision de n dimension. m est le nombre des objectifs à optimiser. Lorsque $m \geq 4$ le problème devient un problème Many-objective.

Dans le cadre de l'optimisation multiobjectifs, les solutions optimales sont également appelées comme solutions non-dominées. Dans un problème de minimisation, une solution x domine une autre solution y lorsque aucune valeur objective de y est inférieure à x et au moins une valeur objective de y est supérieure à x.

La Convergence et la diversité sont les principaux objectifs de tout algorithme d'optimisation multiobjectif. La Convergence se réfère à trouver un ensemble de solutions qui se trouvent sur, ou près, de véritable front Pareto optimale. La diversité vise à trouver un ensemble de solutions qui sont suffisamment diversifiées pour représenter l'ensemble de front Pareto optimale.

5.1.2 Les enjeux d'un problème Many-objective

5.1.2.1 La population non-dominée

La plupart des algorithmes évolutionnaires multiobjectif utilisent le concept de dominance de Pareto afin de comparer et d'identifier les meilleures solutions [51]. Une augmentation du nombre d'objectifs entraîne qu'une grande partie de la population générée de façon aléatoire deviennent non-dominée [51]. Avoir une population qui est en grande partie composée de solutions non-dominées ne donne pas place pour la création de nouvelles solutions à chaque génération et cela ralentit le processus de recherche [26, 87].

5.1.2.2 L'efficacité de calcul

Un autre problème majeur est que les algorithmes d'optimisation Many-objectifs ont tendance à être coûteux en calcul. Certaines matrices de performance tel que l'hypervolume qui exige une puissance de calcul exponentiellement lorsque le nombre d'objectifs augmente, puisque les points de grande dimension sont comparées l'un contre l'autre Pour améliorer ce point, des algorithmes précis et aussi approximatives pour le calcul de l'hypervolume ont été proposées au fil de temps [35].

Les mesures de diversité tel que la distance Crowding deviennent aussi coûteux en calcul lors de l'identification des voisins dans un espace de grande dimension. Pour y remédier, Deb et Jain [35] ont suggéré l'utilisation d'une approche basée sur le point de référence dans lequel les points correspondant à un ensemble de points de référence bien étalés peut trouver un ensemble largement distribué de points Pareto optimales.

5.1.2.3 Recombinaison

Les Opérateurs de recombinaison jouent un rôle important dans les EMO (Evolutionary Multiobjecitve Optimization). Les problèmes surgissent lorsque seulement quelques solutions se trouvent dans un grand espace tridimensionnel que les solutions sont plus susceptibles d'être largement éloignées les unes des autres [35], si l'algorithme de EMO se concentre sur les points extrêmes. Des opérateurs de recombinaison spéciales peuvent être nécessaires pour traiter ce problème. Pour pallier cet problème, Deb et Jain [35] ont suggéré l'utilisation d'un régime spécial de recombinaison dans lequel les solutions parent sont soulignées. Un exemple de ceci est le croisement binaire simulé (SBX) [35, 73]. Une attention particulière doit être mis dans la sélection de l'opérateur et les parents participants.

5.1.2.4 Visualisation

La visualisation est une partie importante de l'optimisation et la prise de décision. Comme le nombre d'objectifs augmente au-delà de trois , il devient difficile pour les chercheurs d'être en mesure de visualiser l'espace objectif. Une manière possible autour de ça serait une visualisation alternée entre les objectifs donnés. Par exemple, un utilisateur peut commencer par le choix d'un ensemble de trois objectifs et de voir la distribution en utilisant un graphique en trois dimensions. L'utilisateur peut alors basculer entre les objectifs et afficher les différents compromis tout en mettant l'accent sur trois objectifs à la fois. Ce sera également une approche efficace dans la prévention de l'utilisateur de

se faire submergé par le grand nombre d'objectifs.

En termes de méthodes graphiques réelles, les coordonnés parallèles [71] ont prouvé d'être un choix populaire dans la résolution de problèmes Many-objective. Une coordonnée parallèle est un tracé à deux dimensions avec l'axe des x représente les objectifs et l'axe y représente les valeurs de la solution. Les caractéristiques géométriques d'une surface en espace à n dimensions sont conservées dans le système de coordonnées parallèles, cependant, cette approche a aussi quelques défauts [48].

5.1.3 Méthodes d'optimisation Many-objective

Des différentes approches ont été explorées pour l'optimisation Many-objectif. Certains ont montré plus de succès que d'autres. Il faut noter que certains de ces méthodes ont été initialement conçues et testées sur deux et trois problèmes objectifs seulement, mais au fil des ans, ils ont été inclus dans des diverses études pour l'optimisation Many-objective.

Un des algorithmes les plus populaires dans la littérature est le NSGA-II [36]. Il est souvent utilisé en tant qu'un algorithme référence de comparaison avec de nouveaux algorithmes. NSGA-II est un MOEA avec un calcul rapide et élitiste basé sur une approche de tri non-dominé. Il utilise également un mécanisme de préservation de diversité explicite pour obtenir un ensemble de solutions Pareto optimales bien réparties. NSGA-II a été initialement testé sur des problèmes avec un plus petit nombre d'objectifs, mais au fil des ans, il a montré pour réussir à résoudre les problèmes avec Many-objectifs.

Un certain nombre d'améliorations de NSGA-II ont été proposées au fil des ans pour faire l'algorithme plus efficace dans le traitement d'un plus grand nombre d'objectifs. ϵ-NSGA-II [79] combine NSGA-II avec un ϵ-dominance archives. Cet algorithme a également été utilisé pour de nombreux problèmes Many-objective du monde réel [79].

HypE (Hypervolume Estimation Algorithm) [9] est un algorithme basé sur l'indicateur hypervolume évolutionnaire multiobjectif rapide qui peut être appliqué à des problèmes avec une nombre arbitraire de fonctions objectifs. Il utilise la simulation Monte-Carlo pour approximer les valeurs de hypervolume exactes. Il a été testé sur un nombre de problèmes Many-objective et a réussi de donner un ensemble concurrentiel des résultats en comparaison avec NSGA-II, SPEA2 et IBEA.

MOEA/D [147] est un autre algorithme qui a montré beaucoup de succès avec l'optimisation Many-objective. Il utilise une méthode de décomposition pour décomposer le problème donné en un certain nombre de problèmes d'optimisation scalaires. Ces sous-

problèmes sont alors optimisés simultanément en utilisant un algorithme évolutionnaire. MOEA/D a été utilisé pour la comparaison dans diverses études récentes [59], ce qui en fait un algorithme de référence pour l'optimisation de Many-objective.

Deb et Jain [35] ont proposé récemment NSGA-III qui utilise une approche basée sur le point de référence pour l'optimisation Many-objective. Le nombre de points de référence est similaire à la taille de la population en sorte que chaque membre de la population est associé à un point de référence. Cela garantit que la diversité des points de référence sont réparties uniformément sur l'hyperplan normalisé. La méthode, qui a été développée spécifiquement pour l'optimisation de Many-objective, a montré des performances supérieures par rapport aux méthodes telles que MOEA/D et NSGA-II. La mise à l'échelle de performance de 15 objectifs a été atteint principalement en raison de l'aide à la préservation de la diversité en fournissant un ensemble de points de référence bien distribués.

5.2 Machine à vecteur de support (SVM)

La méthode de machine à vecteurs de support (Support Vector Machine, SVM) est, dans son origine, binaire. Elle est basée sur le principe de la séparation entre les échantillons des deux classes, l'une positive et l'autre est négative. La méthode SVM est très réussie dans plusieurs domaines d'application compte tenu de la précision qu'elle offre.

5.2.1 Le SVM Binaire

Le SVM Binaire résoudre le problème de séparation de deux classes présentées par n exemples de m attributs. Considérons le problème de faire séparer deux classes représentées par les n exemples :

$$\{(x_1, y_1), \ldots, (x_n, y_n)\}, x_i \in \mathbb{R}^m, y_i \in \{-1, +1\} \tag{5.2}$$

où x_i sont les exemples d'apprentissage et y_i ces classes respectivement. L'objectif de la méthode SVM est de trouver une fonction linéaire f appelée hyperplan qui peut séparer les deux classes :

$$\begin{cases} f(x) = (x \bullet w) + b & \\ f(x) > 0 & \Rightarrow x \in classe + 1 \\ f(x) < 0 & \Rightarrow x \in classe - 1 \end{cases} \tag{5.3}$$

où x est un exemple à classifier, w est un vecteur et b est un biais. Par conséquent, il faut trouver le séparateur qui maximise le plus l'espace entre les marges de deux classes [1].

5.2.2 Le SVM multi-classes

Plusieurs méthodes ont été proposées pour étendre le SVM binaire aux problèmes à plusieurs classes. Chacune de ces méthodes généralisent la capacité de SVM binaires vers un domaine multi-classes. Les méthodes les plus connues sont 1vsR et 1vs1.

5.2.2.1 Un Contre le Reste

Pour chaque k on détermine un hyperplan $H_k(w_k, b_k)$ qui la sépare de toute autres classes, en considérant que cette classe est positive (+1) et les autres classes comme négatives (-1), ce qui résulte, pour un problème de K classes, K SVM binaires.

$$f_k(x) = \langle w_k \bullet x \rangle + b_k \tag{5.4}$$

cette fonction permit de discriminer entre les exemples de la classe k et l'ensemble de toutes autres classes.

5.2.2.2 Un Contre Un

Dans ces méthodes plutôt d'utiliser k fonctions, la méthode 1vs1 (un contre un) fait la discrimination entre chaque classe et chaque autre classe et utilise alors $k(k-1)/2$ fonctions. Par conséquent, l'approche 1vs1 crée un classifieur pour chaque paire de classes.

5.2.3 Le SVM mono-classe

Dans la classification de SVM mono-classe, il est supposé que seuls les échantillons d'une classe, la classe cible, sont disponibles. Cela signifie que seuls les échantillons de la classe cible peuvent être utilisées et aucune information sur d'autres classes est présente. L'objectif de SVM mono-classe est de trouver une frontière entre les échantillons de la classe cible et le reste de l'espace. La tâche consiste à définir une frontière autour de la classe cible, de sorte qu'il accepte autant d'échantillons cibles que possible. Le SVM considère l'origine comme la seule instance de la classe négative, puis tente de trouver un séparateur entre l'origine et des échantillons de la classe cible tout en maximisant la marge entre les deux.

Le problème est de trouver l'hyperplan qui sépare les échantillons cibles de l'origine tout en maximisant la distance entre eux. Le problème est modélisé par le problème de programmation quadratique de l'équation 5.5.

$$\begin{cases} \min_{w,\xi,\rho} \frac{1}{2}||w||^2 + \frac{1}{vN}\sum_{i=1}^{l}\xi_i - \rho \\ \langle w, \phi(x_i)\rangle \geq \rho - \xi_i \\ \xi_i \geq 0 \text{ pour } i = 1, 2, \ldots, l \end{cases} \tag{5.5}$$

où N est le nombre des échantillons de la classe cible, $\langle w, \rho\rangle$ des paramètres pour localiser l'hyperplan, ξ_i c'est l'erreur permis dans la classification des échantillons, pénaliser par les paramètres v, et ϕ est la transformation de l'espace similaire de cas binaire de SVM. Lorsque $\langle w, \rho\rangle$ sont déterminés, tout nouvel échantillon peut être classé par la fonction de décision de l'équation 5.6.

$$f(x) = \langle w, \phi(x_i)\rangle - \rho \tag{5.6}$$

x appartient à la classe cible si $f(x)$ est positive. La résolution de l'équation 5.5 est réalisée en utilisant multiplicateurs de Lagrange pour obtenir le problème dual de l'équation 5.7

$$\begin{cases} \min_{\alpha} & \frac{1}{2}\sum^{i,j} \alpha_j K(x_i, x_j) \\ \text{Avec} & \sum_{i=1}^{n} \alpha_i = 1 \\ & 0 \leq \alpha_i \leq \frac{1}{vl} \end{cases} \tag{5.7}$$

où K est un noyau qui représente l'espace de transformation ϕ. Lorsque les α_i sont déterminées la fonction de décision, pour un échantillon x, est donnée par l'équation 5.8

$$f(x) = \sum_{i=1}^{l} \alpha_i K(x_i, x) - \rho \tag{5.8}$$

où ρ peut être déterminé depuis un échantillon d'apprentissage x_i dont $\alpha_i \neq 0$ par l'équation 5.9

$$\rho = \sum_{j} \alpha_j K(x_j, x_i) \tag{5.9}$$

5.3 Algorithme hybrides CEDA-SVM

Le but de cette proposition est double : (1) pour aider le DM (décideur) de trouver les solutions qui sont à proximité de son intérêt, et (2) pour réduire le nombre d'évaluations de la fonction ce qui diminuent considérablement le temps d'exécution. Pour ce faire, nous utilisons un algorithme à deux phases composé de la phase *Optimisation* , comme il est proposé dans [29], qui trouve un certain nombre de meilleures solutions à un problème défini et la phase *Update* qui trouve un autre ensemble de meilleures solutions jusqu'à ce que le DM est satisfait.

5.3.1 Phase d'Optimisation

Dans cette étape, nous utilisons notre algorithme proposé (Algorithme 11) avec deux étapes (i) la sélection et (ii) la reproduction. Dans l'étape de sélection (interprété par fonction, SelectUsingMOEA), nous utilisons le MOEA/D pour sélectionner les meilleures solutions qui seront utilisées dans l'étape de reproduction où nous faisons usage de Copules pour estimer et régénérer de nouveaux individus.

Algorithme 11 Copula-based EDA

function CEDA
  $\mathbf{P_0} \leftarrow$ Initialisation(m)
  $\mathbf{P} \leftarrow$ SelectUsingMOEA($\mathbf{P_0}$) ▷ En utilisant MOEA/D
  tant que critère d'arrêt n'est pas atteint **faire**
    C $\leftarrow$ ConstructCopulas($\mathbf{P}$)
    $\mathbf{P'} \leftarrow$ GenerateSolutions(C)
    $\mathbf{P''} \leftarrow$ SelectUsingMOEA($[\mathbf{P'P}]^T$) ▷ $[\mathbf{P'P}]^T$ prend tous individus de $\mathbf{P'}$ et $\mathbf{P}$
    $\mathbf{P} \leftarrow \mathbf{P''}$
  fin tant que
  return ($\mathbf{P}, C$)
fin function

Initialement nous mettons population $\mathbf{P_0} = [\mathbf{x_1}, \ldots, \mathbf{x_m}]^T$ où $\mathbf{x_i}, i \in [1, m]$ sont les individus. Chaque individu $\mathbf{x_i} = [x_{i1}, \ldots, x_{in}]$ où $x_{min} \leq x_{ij} \leq x_{max}$. Ensuite, l'étape de sélection utilise les algorithmes MOEA/D - avec Chebyshev ou Boundary Intersection Comme un MOEA. Cette étape retourne un ensemble d'individus, ces individus seront utilisés dans la prochaine étape de l'algorithme qui est l'étape de la reproduction . Nous

appelons $\mathbf{P}$ la matrice des individus résultant du processus de sélection ,pour la première génération nous utilisons la population initiale $\mathbf{P_0}$. Nous avons $\mathbf{P}$ défini comme suit :

$$\mathbf{P} = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix} \tag{5.10}$$

Pour l'étape de reproduction , nous créons les Copules Archimedean, représentées par le vecteur $C = [C_1 \dots C_j \dots C_n]$, en utilisant les sous vecteurs $\mathbf{u_1}, \dots, \mathbf{u_n}$ et $\mathbf{v_1}, \dots, \mathbf{v_n}$, où chaque copule $C_j, j \in [1, n]$ est construite à partir des sous vecteurs $\mathbf{u_j}$ et $\mathbf{v_j}$ comme indiqué dans l'algorithme 8.

Nous utilisons les copules construites $C_1, \dots, C_n$ pour générer de nouveaux individus. L'ensemble des nouveaux individus générés $\mathbf{X'} = [\mathbf{w'_1}, \dots, \mathbf{w'_n}]$ où $\mathbf{w'_j}$ est la concaténation de $\mathbf{u'_j}$ et $\mathbf{v'_j}$ qui sont échantillonnés en utilisant la copule C_j. Notez que le vecteur $\mathbf{w'_j}$ (résultant de la concaténation de $\mathbf{u'_j}$ et $\mathbf{v'_j}$) est d'une taille m' qui n'est pas nécessairement la même taille de la population initiale m. Les nouveaux individus sont donc les vecteurs $\mathbf{x'_i}, i \in [1, m']$ où $\mathbf{X'} = [\mathbf{x'_1}, \dots, \mathbf{x'_{m'}}]^T$. L'algorithme 9 résume ces étapes.

5.3.2 La mise à jour en utilisant un SVM mono-classe

En général, le but des algorithmes de classification consiste à classer un objet inconnu à partir de plusieurs classes prédéfinies. Dans le problème de la classification mono-classe, il est supposé que seules les données de la classe cible est disponible pour l'apprentissage du classificateur, alors que l'ensemble de test comprend des exemples positifs (même classe) et négatifs (autres classes).

Les données utilisées ici d'apprentissage est la solution de Pareto (PS) et tous les individus dans le PS sont placés dans une classe (A), et tous les autres individus ne font pas partie de cette catégorie. Dans le processus de classification nous introduisons les solutions générées en utilisant les copules comme une entrée dans le classificateur (SVM mon-classe) et nous supposons que les individus de la classe (A) serait un ensemble de solution Pareto.

Algorithme 12 SVM classifier

1: **function** CLASSIFY($NDset$,P')
2: Train-mono-class-SVM($NDset$)
3: P''=Classify(P')
4: **return** P''
5: **fin function**

Comme décrit dans les sections précédentes notre proposition vise à donner plus de solutions de rechange au DM dans un très petit temps d'exécution, de sorte que dans la phase d'optimisation, nous calculons de nouvelles solutions, comme indiqué dans l'algorithme 7. Ces solutions peuvent ne pas répondre aux besoins de DM et il aura besoin qu'une autre phase de la nouvelle génération de solutions est nécessaire. La phase de mise à jour proposée dans ce travail (comme indiqué dans l'algorithme 13) rend possible pour le décideur de trouver d'autres nouvelles solutions rapidement en utilisant les Copules construites dans la phase d'optimisation et un SVM mono-classe. Tout d'abord, nous appelons la fonction GenerateSolutions avec C (les copules construites dans la phase d'optimisation), et m'' le nombre de nouvelles solutions nécessaires, puis nous utilisons un SVM mono-classe pour classer les individus générés comme indiqué dans l'algorithme 12. La sortie de l'étape de mise à jour est la population $\mathbf{P}_{\text{update}}$ mise à jour.

Algorithme 13 Update Solutions

1: **function** UPDATESOLUTIONS(C, m'')
2: $\mathbf{P}_{\text{tmp}} \leftarrow$ GenerateSolutions(C, m'')
3: $\mathbf{P}_{\text{update}} \leftarrow$ Classify(ParetoSolutions,$\mathbf{P}_{\text{tmp}}$)
4: **return** $\mathbf{P}_{\text{update}}$
5: **fin function**

5.4 Expérimentation

Dans cette section, la méthode proposée sera comparée avec les algorithmes bien connus : algorithme évolutionnaire multiobjectif basé sur la décomposition (MOEA/D) avec deux méthodes de décomposition (Boundary Intersection (MOEA/D-BI) et Tchebycheff approche (MOEA D-Tchy)). Les expériences sont menées sur quatre problèmes Many-objective benchmark.

5.4.1 Les problèmes benchmarks et les métriques

Pour prouver l'efficacité de l'algorithme proposé, nous avons choisi d'utiliser un ensemble de problèmes de référence, et pour la métrique nous calculons la métrique IGD pour toute référence utilisée, nous avons utilisé les benchmark DTLZ avec des différentes dimensions supérieures ou égale à 3 , les problèmes tests suivantes sont utilisés :

TABLE 5.1: Nombre de générations et dimension pour chaque benchmark

Benchmark	Dimension	Générations
DTLZ(1,2,3,4)	3	250
DTLZ(1,2,3,4)	5	350
DTLZ(1,2,3,4)	8	500
DTLZ(1,2,3,4)	10	750
DTLZ(1,2,3,4)	15	1000

Chaque algorithme est exécuté 20 fois, indépendamment pour chaque occurrence de test, le paramètre de configuration dans le présent document est le suivant :

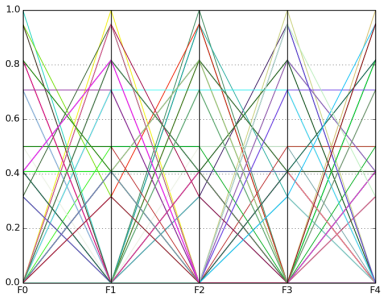
TABLE 5.2: Nombre des individus et la dimension pour chaque benchmark

Dimension	Nombre des individus
3	120
5	126
8	120
10	220
15	120

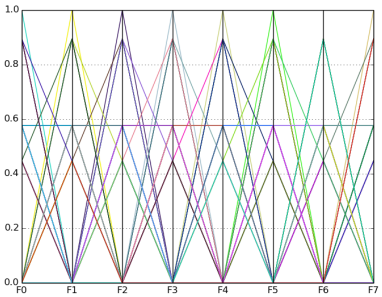
5.4.2 Résultats de simulation

Dans cette section, nous allons montrer les résultats des différentes simulations de l'algorithme, à chaque étape de la simulation en exécutant l'algorithme CEDA utilisant MOEA/D avec TChebyshev et Boundary Intersection sur les différents benchmark. Les tableaux et les chiffres présentés dans les sections suivantes permettront de clarifier les résultats de simulation de l'algorithme proposé.

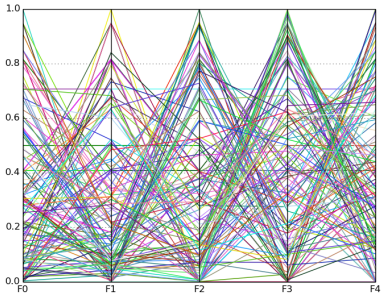
La figure 5.2 illustre le résultat de benchmark DTLZ1 avec 3 dimensions à l'aide MOEA/D-BI et MOEA/D-T (Chebyshev), et il montre que la phase de mise à jour donne plus des solutions alternatives.



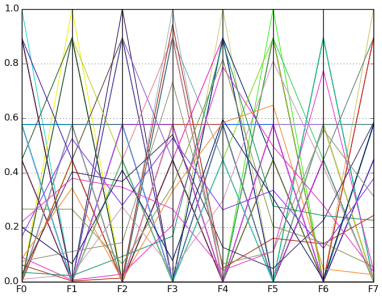
(a) MOEA/D-B fo 5 dimensions



(b) MOEA/D-B for 8 dimensions



(c) CEDA-B for 5 dimensions



(d) CEDA-B 8 dimensions

FIGURE 5.1: Front Pareto de problème DTLZ3 à 5 et 8 objectifs à l'aide du MOEA/D-B, avant et après la mise à jour de front Pareto en utilisant CEDA.

La figure 5.3 illustre le résultat de benchmark DTLZ3 avec 3 dimensions à l'aide MOEA/D-BI et MOEA/D-T (Chebyshev), et il montre que la phase de mise à jour donne plus des solutions alternatives.

Les Figures 5.1a et 5.1c illustrent les résultats de DTLZ3 avec 5 dimensions en utilisant MOEA/D-BI, et il montre que la phase de mise à jour donne plus de solutions de rechange. En outre, les figures 5.1b et 5.1d illustrent les résultats de DTLZ3 avec 8 dimensions où nous utilisons plusieurs objectifs que la précédente benchmark, et il montre que l'utilisation de CEDA phase de mise à jour donne des solutions alternatives de plus.

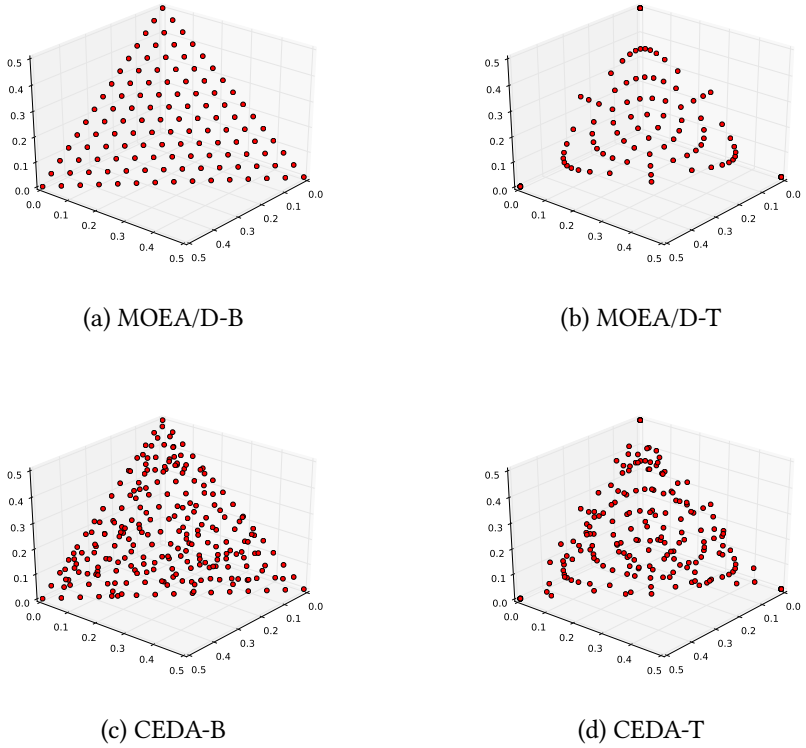
TABLE 5.3: La moyenne de métrique IGD pour les benchmark utilisées avec les différentes dimensions

Benchmark	MOEA/D-B	CEDA-B	MOEA/D-T	CEDA-T
DTLZ1, 3D	$1.66e-02$	$1.38e-02$	$3.10e-02$	$2.04e-02$
DTLZ1, 5D	$1.82e+01$	$1.95e+01$	$2.13e+01$	$2.75e+01$
DTLZ1, 8D	$1.44e-01$	$1.22e-01$	$1.16e-01$	$9.45e-02$
DTLZ1, 10D	$3.77e-02$	$3.68e-02$	$2.35e-01$	$2.33e-01$
DTLZ1, 15D	$6.46e-02$	$6.32e-02$	$3.12e-01$	$3.13e-01$

DTLZ2, 3D	$4.33e-02$	$2.41e-02$	$7.45e-02$	$3.12e-02$
DTLZ2, 5D	$3.63e-01$	$2.82e-01$	$2.14e-01$	$1.51e-01$
DTLZ2, 8D	$4.43e-01$	$3.64e-01$	$4.70e-01$	$4.56e-01$
DTLZ2, 10D	$8.60e-02$	$8.34e-02$	$7.56e-01$	$7.33e-01$
DTLZ2, 15D	$8.67e-02$	$8.84e-02$	$9.43e-01$	$9.17e-01$
DTLZ3, 3D	$4.33e-02$	$3.90e-02$	$7.45e-02$	$5.99e-02$
DTLZ3, 5D	$1.51e+02$	$1.59e+02$	$1.20e+02$	$1.33e+02$
DTLZ3, 8D	$4.46e-01$	$4.25e-01$	$4.59e-01$	$4.46e-01$
DTLZ3, 10D	$1.03e-01$	$9.94e-02$	$8.10e-01$	$8.07e-01$
DTLZ3, 15D	$1.60e-01$	$1.59e-01$	$1.01e+00$	$1.01e+00$
DTLZ4, 3D	$5.12e-02$	$5.08e-02$	$7.45e-02$	$7.12e-02$
DTLZ4, 5D	$7.84e-01$	$7.60e-01$	$2.94e-01$	$2.15e-01$
DTLZ4, 8D	$5.66e-01$	$5.71e-01$	$5.18e-01$	$5.12e-01$
DTLZ4, 10D	$4.58e-01$	$4.53e-01$	$7.75e-01$	$7.76e-01$
DTLZ4, 15D	$4.76e-01$	$4.69e-01$	$9.18e-01$	$9.12e-01$

TABLE 5.4: La moyenne de nombre de nouvelle solution obtenue pour les benchmark utiliseés avec les différentes dimensions.

Benchmark	MOEA/D-B	CEDA-B	MOEA/D-T	CEDA-T
DTLZ1, 3D	96.00	566.08	96.00	559.32
DTLZ1, 5D	96.00	4.16	96.00	11.80
DTLZ1, 8D	95.56	42.12	96.00	149.88
DTLZ1, 10D	173.36	5.12	175.80	4.08
DTLZ1, 15D	92.84	0.12	95.04	1.84
DTLZ2, 3D	96.00	526.04	96.00	555.52
DTLZ2, 5D	96.00	152.56	96.00	638.40
DTLZ2, 8D	96.00	50.72	96.00	68.80
DTLZ2, 10D	176.00	17.80	176.00	38.88
DTLZ2, 15D	96.00	1.68	96.00	37.76
DTLZ3, 3D	96.00	554.68	96.00	673.64
DTLZ3, 5D	96.00	2.08	96.00	6.32
DTLZ3, 8D	95.20	24.68	95.56	166.08
DTLZ3, 10D	172.08	6.00	174.04	27.08
DTLZ3, 15D	88.16	0.00	89.60	0.68
DTLZ4, 3D	96.00	19.08	96.00	34.48
DTLZ4, 5D	96.00	22.60	96.00	114.00
DTLZ4, 8D	96.00	74.08	96.00	28.76
DTLZ4, 10D	176.00	35.00	176.00	20.12
DTLZ4, 15D	96.00	28.72	96.00	7.68



(a) MOEA/D-B (b) MOEA/D-T

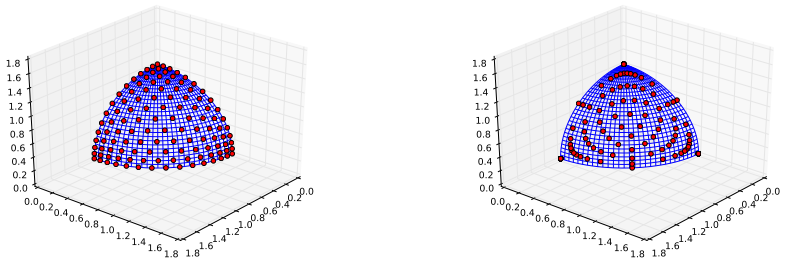
(c) CEDA-B (d) CEDA-T

FIGURE 5.2: front Pareto de problème DTLZ1 avec trois objectifs en utilisant le MOEA/D-B, MOEA/D-T, avant et après la mise à jour de front Pareto en utilisant CEDA.

Conclusion

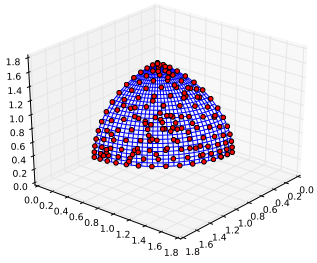
Dans ce chapitre, nous avons présenté notre algorithme hybride CEDA-SVM. L'algorithme proposé utilise une méthode d'estimation très utile dans les statistiques qui est la copule. Ensuite, nous avons montré la capacité de l'algorithme CEDA-SVM à résoudre des problèmes d'optimisation Many-objective puis l'utilisation des solutions Pareto obtenus pour générer des nouvelles Solutions Pareto dans une manière très efficace en utilisant les SVM. La nouvelle solution générée peut enrichir l'espace de choix de la décision d'un décideur.

L'algorithme CEDA-SVM proposé utilise dans la phase d'estimation une copule. Cette copule est le modèle utilisé pour décrire la distribution et des dépendances entre les solutions Pareto obtenues en une génération de l'algorithme. La copule est créée en considérant que l'ensemble des solutions Pareto ont une sorte de dépendance. Les nouveaux individus seront générés à l'aide de la copule, ces individus sont classés à l'aide d'un SVM mon-classe et les meilleurs sont sélectionnés. L'algorithme proposé montre de bons ré-

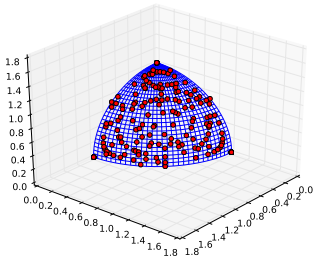


(a) MOEA/D-B

(b) MOEA/D-T



(c) CEDA-B



(d) CEDA-T

FIGURE 5.3: front Pareto de problème DTLZ3 avec trois objectifs en utilisant le MOEA/D-B, MOEA/D-T, avant et après la mise à jour de front Pareto en utilisant CEDA.

sultats avec les quatre problèmes de référence Many-objective utilisés.

Chapitre 6

Expérimentation

Introduction

Dans ce chapitre on va expliquer en détail les problèmes utilisés pour évaluer les différentes qualités de l'algorithme proposé. Ces problèmes sont généralement utilisés lorsque on vise à tester des nouveaux algorithmes dans le domaine de l'optimisation multiobjectif.

Après avoir testé l'algorithme sur un problème donné il faut ensuite évaluer la qualité des résultats produit en utilisant des métriques de qualité. Dans le domaine de l'optimisation multiobjectif, les principaux caractéristiques qu'une métrique doit les mesurer sont la *Diversité* des solutions toute au long de front Pareto, et la *Convergence* des solutions approximées vers le front Pareto optimal.

Ce chapitre commence par une explication des métriques utilisées pour évaluer les solutions produites par l'exécution des problèmes tests, la diversité en détaillant sur la métrique Δ, la métrique GD [15, 142, 144, 157] et IGD, les métriques de convergence, ensuite les différentes classes de problèmes tests utilisés . La deuxième partie de ce chapitre concerne les résultats obtenus avec les différentes qualités de ces résultats.

6.1 Métriques de performance

Au lieu de trouver une seule solution optimale comme dans l'optimisation monoobjectif, l'optimisation multiobjectif vise à trouver un ensemble de solutions de compromis qui répondent à quatre objectifs communs de l'optimisation multiobjectifs, notamment la proximité, la diversité, l'uniformité, et le nombre de solutions non dominées.

Dans ce cas, des mesures de performance sont des outils importants qui fournissent une mesure quantitative scalaire de l'ensemble généré de solutions.

6.1.1 La métrique Δ (Spread)

Dans les MOPs, l'intérêt principal est de parvenir à un ensemble de solutions qui couvre toute la région Pareto optimale. La mesure de la propagation (Δ) suggérée par Deb [39] mesure le degré de la propagation atteint parmi les solutions obtenues. En outre, La métrique consiste à calculer la non-uniformité de la distribution :

$$\Delta = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{|Q|} |d_i - \overline{d}|}{\sum_{m=1}^{M} d_m^e + |Q|\overline{d}} \tag{6.1}$$

où les d_i sont des distances euclidiennes entre solutions voisins ayant la valeur moyenne $\overline{d}$. Le paramètre d_m^e est la distance entre les solutions extrêmes de P^* (le vrai front Pareto optimal) et Q correspondant à le m-ème fonction objectif. Un algorithme capable de trouver une valeur plus faible de Δ est en mesure de trouver un ensemble plus diversifié de solutions non-dominées.

6.1.2 Distance générational GD

Le métrique GD est défini comme suit :

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i}}{n} \tag{6.2}$$

où $d_i = \min_j ||f(x_i) - PF_{true}(x_j)||$ correspond à la distance dans l'espace objectif entre l'individu x_i et le membre le plus proche dans le vrai front Pareto, et n est le nombre des individus dans le front approximé. Ce métrique, est une mesure représentant jusqu'à quel point le front approximé est "loin" du vrai front Pareto. Une valeur inférieure de GD représente une meilleure performance.

6.1.3 Distance générational inversée IGD

Cette métrique mesure à la fois la convergence et la diversité. Soit PF_{true} un ensemble de solutions uniformément réparties dans le vrai front Pareto. X est l'ensemble des solutions non-dominées dans le front approximé PF_{known}

$$IGD = \frac{\sum_{v \in PF_{true}} d(v, X)}{|PF_{true}|} \tag{6.3}$$

$d(v, X)$ désigne la distance euclidienne minimale entre v et les points de X. Pour avoir une faible valeur de IGD , l'ensemble X devrait être proche de PF_{true} et ne peut pas manquer une partie de l'ensemble du PF_{true} [158].

6.1.4 Hypervolume H

Ce métrique calcule l'hypervolume de l'espace objectif multidimensionnel enfermé par le front approximé PF_{known} et un point de référence. Par exemple, un individu x_i dans PF_{known} pour un MOP de deux dimensions définit une zone rectangle, $a(x_i)$, limitée par une origine et $f(x_i)$. L'union de ces secteurs de rectangle est appelé "hyperarea" de PF_{known}

$$H(PF_{known}) = \left\{ \bigcup_i a(x_i) | \forall x_i \in PF_{known} \right\} \tag{6.4}$$

Cette mesure nécessite de définir un point de référence de la région et pourrait être trompeuse si PF_{known} est non-convexe [158].

6.2 Les problèmes test

Pour évaluer l'efficacité de notre algorithme proposé, nous avons choisi de le tester sur un ensemble de problèmes de référence généralement utilisés dans la littérature. Plus précisément, nous utilisons les problèmes de référence Poloni,Kursawe ,Fonseca,Schaffer, problèmes ZDT, problèmes UF,CF définis dans CEC2009 [150], pour la première contribution (optimisation multiobjectif) et les DTLZ pour les problèmes Many-objective. Nous opérons sur 100 individus et définir le nombre maximum de l'évaluation à 300000. Nous varions le nombre d'appels de DM et de montrer les résultats obtenus avec 5 et 20 appels de DM.

Poloni :

$$\begin{aligned}
&f_1(x,y) = \left[1 + (A_1 - B_1(x,y))^2 + (A_2 - B_2(x,y))^2\right] \\
&f_2(x,y) = (x+3)^2 + (y+1)^2 \\
&A_1 = 0.5\sin(1) - 2\cos(1) + \sin(2) - 1.5\cos(2) \\
&A_2 = 1.5\sin(1) - \cos(1) + 2\sin(2) - 0.5\cos(2) \\
&B_1(x,y) = 0.5\sin(x) - 2\cos(x) + \sin(y) - 1.5\cos(y) \\
&B_2(x,y) = 1.5\sin(x) - \cos(x) + 2\sin(y) - 0.5\cos(y) \\
&\text{Avec } x, y \in [-\pi, \pi]
\end{aligned} \tag{6.5}$$

Kursawe :

$$\begin{aligned}
&f_1(x) = \textstyle\sum_{i=1}^{n-1}\left[-10e^{-0.2\sqrt{x_i^2+x_{i+1}^2}}\right] \\
&f_2(x) = \textstyle\sum_{i=1}^{n}\left[|x_i|^{0.8} + 5\sin(x_i)^3\right] \\
&\text{Avec } x \in [-5,5] \text{ où } x \in R^3
\end{aligned} \tag{6.6}$$

Fonseca :

$$\begin{aligned}
&f_1(\mathbf{x}) = 1 - e^{-\sum_{i=1}^{3}(x_i - \frac{1}{\sqrt{3}})^2} \\
&f_2(\mathbf{x}) = 1 - e^{-\sum_{i=1}^{3}(x_i + \frac{1}{\sqrt{3}})^2} \\
&\text{Avec } x \in [-4,4]
\end{aligned} \tag{6.7}$$

Schaffer :

$$\begin{aligned}
&f_1(\mathbf{x}) = x_1^2 \\
&f_2(\mathbf{x}) = (x_1 - 2)^2 \\
&\text{Avec } x \in [-10^3, 10^3]
\end{aligned} \tag{6.8}$$

6.2.1 Les problèmes test ZDT

Deb a identifié plusieurs caractéristiques qui peuvent provoquer des difficultés pour les EAs multiobjectif en 1) convergeant vers le front Pareto optimal et 2) le maintien de la diversité au sein de la population. Concernant la première question, la multimodalité, la déception, et le problème à optimal isolés sont des problèmes bien connus dans l'optimisation évolutionnaire mono-objectif. La deuxième question est importante pour parvenir à un front non-dominé bien réparti. Cependant, certaines caractéristiques du front Pareto optimal peuvent empêcher un EA de trouver des solutions Pareto optimales diverses : la convexité ou non-convexité, discret et non-uniformité. Pour chacune des six problèmes à caractéristiques mentionnées , une fonction de test correspondante est construite suivant les directives de Deb [155].

Chacune des fonctions de test définies est structurée de la même manière et se com-

pose de trois fonctions f_1, g, h [155].

$$\begin{array}{lrcl} \text{Minimiser} & \mathcal{T}(\mathbf{x}) & = & (f_1(x_1), f_2(\mathbf{x}) \\ \text{avec} & f_2(\mathbf{x}) & = & g(x_2, \ldots, x_m)h(f_1(x_1), g(x_2, \ldots, x_m)) \\ \text{où} & \mathbf{x} & = & (x_1, \ldots, x_m) \end{array} \tag{6.9}$$

La fonction f_1 est une fonction de la première variable de décision uniquement, g est une fonction des $m-1$ autres variables, et les paramètres de h sont les valeurs de la fonction de f_1 et g. Les fonctions de test se diffèrent dans ces trois fonctions, ainsi que le nombre de variables m et les valeurs dans les variables.

Nous introduisons les six fonctions de test $\mathcal{T}_1, \ldots, \mathcal{T}_6$ qui suivent le schéma donné dans l'équation 6.9.

ZDT1 : La fonction test $\mathcal{T}_1$ a un front de Pareto-optimal convexe :

$$\begin{array}{lcl} f_1(x_1) & = & x_1 \\ g(x_2, \ldots, x_m) & = & 1 + 9 \cdot \sum_{i=2}^{m} x_i/(m-1) \\ h(f_1, g) & = & 1 - \sqrt{f_1/g} \end{array} \tag{6.10}$$

où $x_i \in [0, 1]$ et $m = 30$. Le front Pareto optimal est formé avec $g(\mathbf{x}) = 1$.

ZDT2 : La fonction test $\mathcal{T}_2$ est la contrepartie non-convexe à $\mathcal{T}_1$:

$$\begin{array}{lcl} f_1(x_1) & = & x_1 \\ g(x_2, \ldots, x_m) & = & 1 + 9 \cdot \sum_{i=2}^{m} x_i/(m-1) \\ h(f_1, g) & = & 1 - (f_1/g)^2 \end{array} \tag{6.11}$$

où $x_i \in [0, 1]$ et $m = 30$. Le front Pareto optimal est formé avec $g(\mathbf{x}) = 1$.

ZDT3 : La fonction test $\mathcal{T}_3$ représente la caractéristique de discrétisation ; son front Pareto optimal se compose de plusieurs parties convexes non-contigues :

$$\begin{array}{lcl} f_1(x_1) & = & x_1 \\ g(x_2, \ldots, x_m) & = & 1 + 9 \cdot \sum_{i=2}^{m} x_i/(m-1) \\ h(f_1, g) & = & 1 - \sqrt{f_1/g} - (f_1/g)sin(10\pi f_1) \end{array} \tag{6.12}$$

où $x_i \in [0, 1]$ et $m = 30$. Le front Pareto optimal est formé avec $g(\mathbf{x}) = 1$. L'introduction de la fonction sinus en h provoque une discontinuité dans le front Pareto optimal. Cependant, il n'y a pas de discontinuité dans l'espace des paramètres.

ZDT4 : La fonction test $\mathcal{T}_4$ contient 21^9 fronts Pareto optimaux locaux et, par consé-

quent, des tests de la capacité de EA pour faire face à la multimodalité :

$$\begin{array}{lcl} f_1(x_1) & = & x_1 \\ g(x_2,\ldots,x_m) & = & 1+10(m-1)+\sum_{i=2}^{m}(x_i^2-10\cos(4\pi x_i)) \\ h(f_1,g) & = & 1-\sqrt{f_1/g} \end{array} \tag{6.13}$$

où $m = 10$ et $x_1 \in [0,1]$, $x_2,\ldots,x_m \in [-5,5]$. Le front Pareto optimal global est formé avec $g(\mathbf{x}) = 1$, le meilleur front Pareto optimal local avec $g(\mathbf{x}) = 1.25$. Notez que tous les ensembles Pareto optimaux locaux se distinguent dans l'espace objectif.

ZDT5 : La fonction test $\mathcal{T}_5$ décrit un problème trompeur et se distingue des autres fonctions de test dans le fait que x_i représente une chaîne binaire

$$\begin{array}{lcl} f_1(x_1) & = & 1+u(x_1) \\ g(x_2,\ldots,x_m) & = & \sum_{i=2}^{m} v(u(x_i)) \\ h(f_1,g) & = & 1/f_1 \end{array} \tag{6.14}$$

où $u(x_i)$ donne le nombre des uns dans le vecteur binaire x_i,

$$v(u(x_i)) = \left\{ \begin{array}{ll} 2+u(x_i) & si \quad u(x_i) < 5 \\ 1 & si \quad u(x_i) = 5 \end{array} \right\}$$

et $m = 11$, $x_1 \in \{0,1\}^{30}$ et $x_2,\ldots,x_m \in \{-5,5\}^5$. Le vrai front Pareto optimale est formé avec $g(\mathbf{x}) = 10$, tandis que le meilleur front Pareto-optimale trompeuse est représenté par la solution où $g(\mathbf{x}) = 11$. Le front Pareto optimale globale ainsi que celles locales sont convexes.

ZDT6 : La fonction test $\mathcal{T}_6$ comprend deux difficultés causées par la non-uniformité de l'espace de recherche : d'abord, les solutions Pareto optimales sont non-uniformément réparties le long du front Pareto global ; d'autre part, la densité des solutions est plus faible près du front Pareto optimal et plus grand loin du front.

$$\begin{array}{lcl} f_1(x_1) & = & 1-exp(-4x_1)\sin^6(6\pi x_1) \\ g(x_2,\ldots,x_m) & = & 1+9\cdot((\sum_{i=2}^{m} x_i)/(m-1))^{0.25} \\ h(f_1,g) & = & 1-(f_1/g)^2 \end{array} \tag{6.15}$$

où $m = 10$ et $x_i \in [0,1]$. Le front Pareto optimal est formé avec $g(\mathbf{x}) = 1$ et il est non-convexe.

6.2.2 Les problèmes test DTLZ

Problème test DTLZ1 : Comme un problème de test simple, nous construisons un problème M-Objectif avec un front Pareto optimal linéaire :

$$\begin{array}{rlcl} \text{Minimiser} & f_1(\mathbf{x}) & = & \frac{1}{2}x_1x_2\cdots x_{M-1}(1+g(\mathbf{x}_M)) \\ \text{Minimiser} & f_2(\mathbf{x}) & = & \frac{1}{2}x_1x_2\cdots(1-x_{M-1})(1+g(\mathbf{x}_M)) \\ & \vdots & \vdots & \\ \text{Minimiser} & f_{M-1}(\mathbf{x}) & = & \frac{1}{2}x_1(1-x_2)(1+g(\mathbf{x}_M)) \\ \text{Minimiser} & f_M(\mathbf{x}) & = & \frac{1}{2}(1-x_1)(1+g(\mathbf{x}_M)) \\ \text{Avec} & 0 \leq x_i \leq 1 & , & i=1,2,\cdots,n \end{array} \tag{6.16}$$

La dernière $k = (n - M + 1)$ des variables sont représentées comme $\mathbf{x}_M$. La fonction $g(\mathbf{x}_M)$ exige $|\mathbf{x}_M| = k$ des variables et doit prendre toute fonction avec $g \geq 0$. Nous suggérons ce qui suit :

$$g(\mathbf{x}_M) = 100\left[|\mathbf{x}_M| + \sum_{x_i \in \mathbf{x}_M}(x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))\right] \tag{6.17}$$

Les solutions Pareto optimales correspond à $x_i = 0,5$ (pour tout $x_i \in \mathbf{x}_M$) et les valeurs de la fonction objectif se trouvent sur l'hyperplan linéaire : $\sum_{m=1}^{M} f_m^* = 0.5$. La seule difficulté fournie par ce problème est la convergence à l'hyperplan Pareto optimal. L'espace de recherche contient $(11^k - 1)$ fronts Pareto optimales locales, où un MOEA peut être attiré avant d'atteindre le front Pareto-optimal global [40].

Problème test DTLZ2 :

$$\begin{array}{rlcl} \text{Minimiser} & f_1(\mathbf{x}) & = & (1+g(\mathbf{x}_M))\cos(x_1\pi/2)\cdots\cos(x_{M-2}\pi/2)\cos(x_{M-1}\pi/2) \\ \text{Minimiser} & f_2(\mathbf{x}) & = & (1+g(\mathbf{x}_M))\cos(x_1\pi/2)\cdots\cos(x_{M-2}\pi/2)\sin(x_{M-1}\pi/2) \\ \text{Minimiser} & f_3(\mathbf{x}) & = & (1+g(\mathbf{x}_M))\cos(x_1\pi/2)\cdots\sin(x_{M-2}\pi/2) \\ & \vdots & \vdots & \\ \text{Minimiser} & f_M(\mathbf{x}) & = & (1+g(\mathbf{x}_M))\sin(x_1\pi/2) \\ \text{Avec} & g(\mathbf{x}_M) & = & \sum_{x_i\in\mathbf{x}_M}(x_i-0.5)^2, \\ & 0 \leq x_i \leq 1 & , & i=1,2,\cdots,n \end{array} \tag{6.18}$$

Le vecteur $\mathbf{x}$ est construit avec $k = n-M+1$ variables. Les solutions Pareto optimale correspond à $x_i = 0.5$ pour tous les $x_i \in \mathbf{x}_M$.

Problème test DTLZ3 : Afin d'étudier la capacité d'un MOEA à converger vers le front Pareto optimal global, nous suggérons d'utiliser le problème ci-dessus avec la fonction g donnée dans l'équation 6.17 :

$$\begin{array}{rlcl}
\text{Minimiser} & f_1(\mathbf{x}) & = & (1+g(\mathbf{x}_M))\cos(x_1\pi/2)\cdots\cos(x_{M-2}\pi/2)\cos(x_{M-1}\pi/2) \\
\text{Minimiser} & f_2(\mathbf{x}) & = & (1+g(\mathbf{x}_M))\cos(x_1\pi/2)\cdots\cos(x_{M-2}\pi/2)\sin(x_{M-1}\pi/2) \\
\text{Minimiser} & f_3(\mathbf{x}) & = & (1+g(\mathbf{x}_M))\cos(x_1\pi/2)\cdots\sin(x_{M-2}\pi/2) \\
 & \vdots & \vdots & \\
\text{Minimiser} & f_M(\mathbf{x}) & = & (1+g(\mathbf{x}_M))\sin(x_1\pi/2) \\
\text{Avec} & g(\mathbf{x}_M) & = & 100\left[|\mathbf{x}_M| + \sum_{x_i\in\mathbf{x}_M}(x_i-0.5)^2 - \cos(20\pi(x_i-0.5))\right], \\
 & 0\leq x_i\leq 1 & , & i=1,2,\cdots,n
\end{array} \tag{6.19}$$

La fonction g ci-dessus introduit (3^k-1) fronts de Pareto optimales locales et un front Pareto optimal global. Tous les fronts Pareto optimales locales sont parallèles au front Pareto optimal global et un MOEA peuvent se coincer à l'un de ces fronts Pareto optimales locales, avant de converger vers le front Pareto optimal global (à $g^*=0$). Le front Pareto optimale globale correspond à $x_i=0.5$ pour $x_i\in\mathbf{x}_M$. Le prochain front Pareto optimal local est à $g^*=1$ [40].

Problème test DTLZ4 : Afin d'étudier la capacité d'un MOEA de maintenir une bonne distribution de solutions, nous modifions problème DTLZ2 avec un mappage de variables paramétriques différentes :

$$\begin{array}{rlcl}
\text{Minimiser} & f_1(\mathbf{x}) & = & (1+g(\mathbf{x}_M))\cos(x_1^\alpha\pi/2)\cdots\cos(x_{M-2}^\alpha\pi/2)\cos(x_{M-1}^\alpha\pi/2) \\
\text{Minimiser} & f_2(\mathbf{x}) & = & (1+g(\mathbf{x}_M))\cos(x_1^\alpha\pi/2)\cdots\cos(x_{M-2}^\alpha\pi/2)\sin(x_{M-1}^\alpha\pi/2) \\
\text{Minimiser} & f_3(\mathbf{x}) & = & (1+g(\mathbf{x}_M))\cos(x_1^\alpha\pi/2)\cdots\sin(x_{M-2}^\alpha\pi/2) \\
 & \vdots & \vdots & \\
\text{Minimiser} & f_M(\mathbf{x}) & = & (1+g(\mathbf{x}_M))\sin(x_1^\alpha\pi/2) \\
\text{Avec} & g(\mathbf{x}_M) & = & \sum_{x_i\in\mathbf{x}_M}(x_i-0.5)^2, \\
 & 0\leq x_i\leq 1 & , & i=1,2,\cdots,n
\end{array} \tag{6.20}$$

Le paramètre $\alpha=100$ est utilisé ici. Cette modification permet à un ensemble dense de solutions de se placer près du f_M-f_1 plan [40].

Problème test DTLZ5 :

$$\begin{array}{rlcl}
\text{Minimiser} & f_1(\mathbf{x}) & = & (1+g(\mathbf{x}_M))\cos(\theta_1\pi/2)\cdots\cos(\theta_{M-2}\pi/2)\cos(\theta_{M-1}\pi/2) \\
\text{Minimiser} & f_2(\mathbf{x}) & = & (1+g(\mathbf{x}_M))\cos(\theta_1\pi/2)\cdots\cos(\theta_{M-2}\pi/2)\sin(\theta_{M-1}\pi/2) \\
\text{Minimiser} & f_3(\mathbf{x}) & = & (1+g(\mathbf{x}_M))\cos(x_1\pi/2)\cdots\sin(x_{M-2}\pi/2) \\
& \vdots & \vdots & \\
\text{Minimiser} & f_M(\mathbf{x}) & = & (1+g(\mathbf{x}_M))\sin(\theta_1\pi/2) \\
\text{Avec} & \theta_i & = & \frac{\pi}{4(1+g(\mathbf{x}_M)}(1+2g(\mathbf{x}_M)x_i), \text{Pour } i=2,3,\ldots,(M-1) \\
& g(\mathbf{x}_M) & = & \sum_{x_i\in\mathbf{x}_M}(x_i-0.5)^2, \\
& 0\leq x_i\leq 1 & , & i=1,2,\cdots,n
\end{array} \tag{6.21}$$

Le front Pareto optimal correspond à $x_i = 0.5$ pour tous les $x_i \in \mathbf{x}_M$ et des valeurs de fonction satisfont $\sum_{m=1}^{M} f_m^2 = 1$. Ce problème permettra de tester la capacité d'un MOEA à converger vers une courbe et permettra également un moyen plus facile pour visuellement démontrer (juste en traçant f_M avec toute autre fonction objectif) la performance d'un MOEA [40].

Problème test DTLZ6 : Le problème de test ci-dessus peut être rendu plus difficile en faisant une modification similaire à la fonction g dans DTLZ5, comme cela se fait dans DTLZ3. Toutefois, dans DTLZ6, nous utilisons , une fonction g différente :

$$g(\mathbf{x}_M) = \sum_{x_i\in\mathbf{x}_M} x_i^{0.1} \tag{6.22}$$

Ici, le front Pareto optimal correspond à $x_i = 0$ pour tout $x_i \in \mathbf{x}_M$. La taille de vecteur $\mathbf{x}_M$ est choisi en tant que 10 et le nombre total de variables est identique comme dans DTLZ5 [40].

Problème test DTLZ7 : Ce problème a un ensemble déconnecté des régions Pareto optimale :

$$\begin{array}{rlcl}
\text{Minimiser} & f_1(\mathbf{x}) & = & x_1 \\
\text{Minimiser} & f_2(\mathbf{x}) & = & x_2 \\
& \vdots & \vdots & \\
\text{Minimiser} & f_{M-1}(\mathbf{x}_{M-1}) & = & x_{M-1} \\
\text{Minimiser} & f_M(\mathbf{x}) & = & (1+g(\mathbf{x}_M))h(f_1,f_2,\ldots,f_{M-1},g) \\
\text{où} & g(\mathbf{x}_M) & = & 1+\frac{g}{|\mathbf{x}_M|}\sum_{x_i\in\mathbf{x}_M} x_i \\
& h(f_1,f_2,\ldots,f_{M-1},g) & = & M-\sum_{i=1}^{M-1}\left[\frac{f_i}{1+g}(1+\sin(3\pi f_i))\right] \\
\text{avec} & 0\leq x_i\leq 1 & , & i=1,2,\cdots,n
\end{array} \tag{6.23}$$

Ce problème de test a 2^{M-1} régions Pareto optimale déconnectées dans l'espace de recherche. La fonction g nécessite $k = |x_M| = n - M + 1$ variables de décision. Les solutions Pareto optimale correspond à $\mathbf{x}_M = 0$. Ce problème permettra de tester la capacité d'un algorithme de maintenir les sous-population dans différentes régions Pareto optimales [40].

Problème test DTLZ8 :

$$\begin{array}{rlcl} \text{Minimiser} & f_j(\mathbf{x}) & = & \frac{1}{\lfloor \frac{n}{M} \rfloor} \sum_{i=\lfloor (j-1)\frac{n}{M} \rfloor}^{\lfloor j \frac{n}{M} \rfloor} x_i, j = 1, 2, \ldots, M \\ \text{Avec} & g_j(\mathbf{x}) & = & f_M(\mathbf{x}) + 4 f_j(\mathbf{x}) - 1 \geq 0, j = 1, 2, \ldots, (M-1) \\ & g_M(\mathbf{x}) & = & 2 f_M(\mathbf{x}) + \min_{\substack{i,j=1 \\ i \neq j}}^{M-1} [f_i(\mathbf{x}) + f_j(\mathbf{x})] - 1 \geq 0 \\ & 0 \leq x_i \leq 1 & , & i = 1, 2, \cdots, n \end{array} \tag{6.24}$$

Ici, le nombre de variables est considéré comme étant plus grand que le nombre d'objectifs ou $n > M$. Nous suggérons $n = 10M$. Dans ce problème, il y a un total de M contraintes. Le front Pareto optimale est une combinaison d'une ligne droite et un hyperplan. La ligne droite est l'intersection de la $(M-1)$ première contraintes (avec $f_1 = f_2 = \cdots = f_{M-1}$) et l'hyperplan est représenté par la contrainte g_M. Les MOEAs peuvent avoir des difficultés pour trouver des solutions dans les deux régions dans ce problème et également à maintenir une bonne distribution de solutions sur l'hyperplan [40].

Problème test DTLZ9 :

$$\begin{array}{rlcl} \text{Minimiser} & f_j(\mathbf{x}) & = & \sum_{i=\lfloor (j-1)\frac{n}{M} \rfloor}^{\lfloor j \frac{n}{M} \rfloor} x_i^{0.1}, j = 1, 2, \ldots, M \\ \text{Avec} & g_j(\mathbf{x}) & = & f_M^2(\mathbf{x}) + f_j^2(\mathbf{x}) - 1 \geq 0, j = 1, 2, \ldots, (M-1) \\ & 0 \leq x_i \leq 1 & , & i = 1, 2, \cdots, n \end{array} \tag{6.25}$$

Le nombre de variables est considéré comme étant plus grand que le nombre d'objectifs. Pour ce problème, nous utilisons également $n = 10M$. Le front Pareto optimal est une courbe avec $f_1 = f_2 = \cdots = f_{M-1}$, similaire à celle de DTLZ5. Cependant, la densité de solutions devient plus fine vers la région Pareto optimal [40].

6.2.3 Métriques de qualité proposée

En plus d'examiner les indicateurs traditionnellement utilisés pour évaluer la qualité des solutions obtenues, à savoir la diversité et de la convergence, nous définissons une nouvelle mesure, la vitesse de régénération, pour mesurer comment de nouvelles solutions peuvent être obtenues rapidement. La nouvelle mesure permet de montrer l'effi-

cacité de notre algorithme qui permet de trouver de nouvelles solutions rapidement et sans compromettre leurs qualités.

La métrique vitesse de mise à jour des solutions, mesure le nombre de nouvelles solutions obtenues sur une période de temps (exprimée en termes de nombre d'évaluations des fonctions objectifs), comme indiqué dans (6.26).

Pour montrer le nouvel aspect garanti par notre algorithme qui est la capacité à obtenir de nouveaux PS avec un temps très petit (négligeable), nous avons proposé une métrique qui calcule le nombre de différents solutions Pareto entre deux ensembles de PS que nous pouvons définir comme suit :

$$I_{\text{new}} = \frac{\sum_{t=0}^{T} |PS_t|}{\sum_{t=0}^{T} FE_t} \tag{6.26}$$

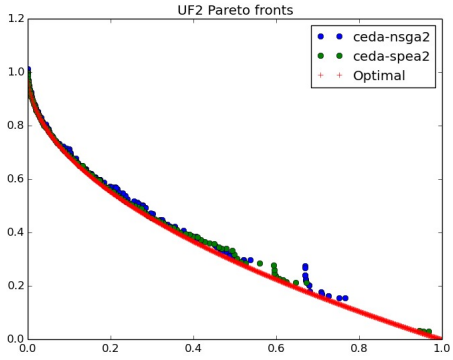
où $|PS_t|$ représente le nombre de solutions Pareto obtenues à l'itération t, $|FE_t|$ est le nombre des évaluations de fonction, et T est le nombre d'itérations (le nombre de fois où le décideur appelle de nouveau l'algorithme pour trouver de nouvelles solutions).

6.3 Les résultats de Simulation

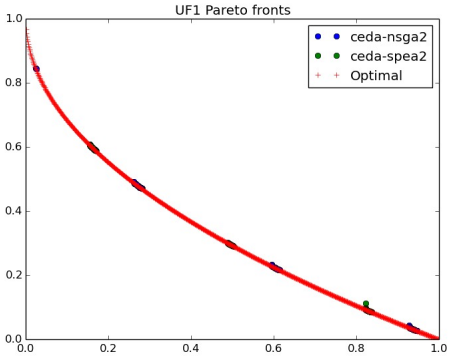
Dans cette section, nous montrons que notre méthode proposée CEDA atteint une bonne diversité et convergence par rapport à celles obtenues avec les méthodes de l'état de l'art tels que SPEA2 et NSGA2 en considérant les problèmes de référence utilisés tels que (UF1, ..., UF10 et CF1, ... , CF10) de la compétition CEC2009[150] et autres.

6.3.1 Qualités de Solutions (Stage d'optimisation)

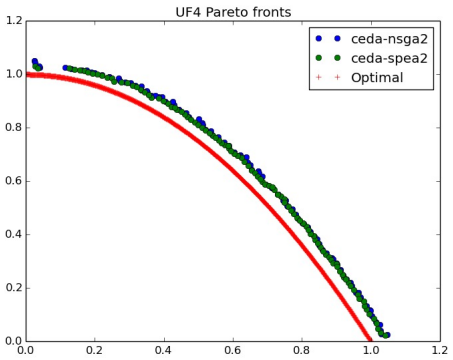
Dans Les figures 6.2, 6.3, nous montrons les Fronts Pareto obtenus lors de la résolution des problèmes de test considérés avec notre méthode proposée CEDA-SPEA2 (CEDA avec SPEA2 comme méthode de sélection) et CEDA-NSGA2 (CEDA avec NSGA2 comme méthode de sélection). Nous montrons que les deux algorithmes CEDA trouvent des solutions ayant des qualités similaires indépendamment de l'algorithme utilisé pour la sélection (ou SPEA2 NSGA2), parce que les solutions sont générées selon la même technique à base de copules - SPEA2 ou NSGA2 ne sont utilisés que pour la sélection.



(a) UF2



(b) UF1



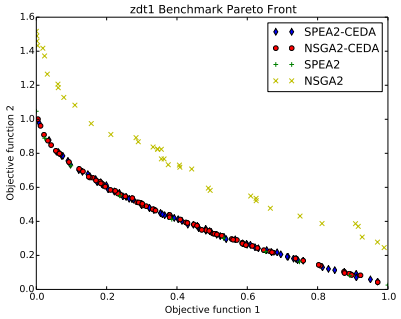
(c) UF4

FIGURE 6.1: Front de Pareto pour les benchmarks UF1,UF2,UF4 en utilisant CEDA-NSGA2

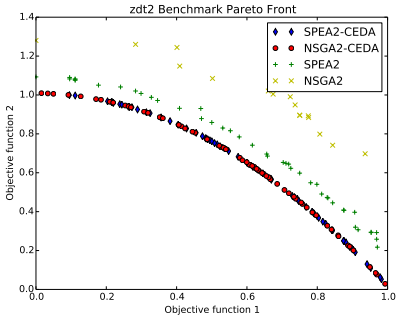
Figure 6.1 montre que CEDA-NSGA2 et CEDA-SPEA2 fournissent des solutions avec des qualités différentes sur les benchmarks UF2 et UF1 car ils utilisent des différentes techniques pour trouver de nouvelles solutions. Figure 1 et la Figure 2 montrent également que notre proposition offre toujours des solutions qui sont à la proximité de front optimale, en particulier sur les problèmes UF5, UF4 et CF1.

La figure 6.4 montre que l'algorithme CEDA proposé génère plus des solutions Pareto par rapport à ceux obtenus par SPEA2 traditionnel. Des résultats similaires ont été obtenus avec CEDA par rapport à NSGA2 traditionnel.

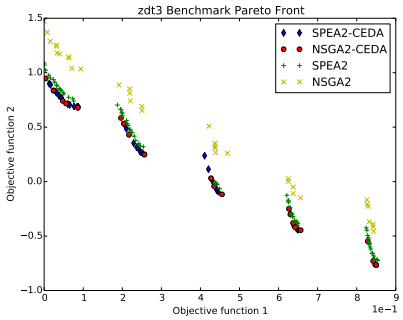
Nous montrons que les deux variantes de CEDA convergent au front optimal de Pareto d'une manière similaire à NSGA2 et SPEA2. Cela montre que la copule est très bonne et comparable aux opérateurs génétiques classiques (mutation et de croisement) en termes de reproduction.



(a) ZDT1



(b) ZDT2



(c) ZDT3

FIGURE 6.2: Front de Pareto pour les benchmarks ZDT1, ZD2, ZDT3, en utilisant CEDA-NSGA2, CEDA-SPEA2 comme méthode de sélection pour CEDA comparé avec l'utilisation de NSGA2 et SPEA2

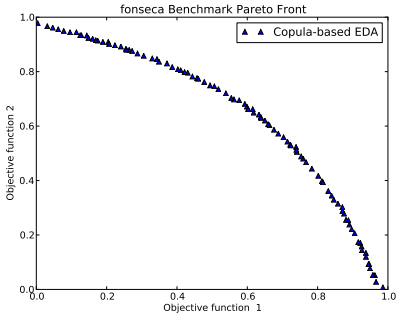
6.3.2 Convergence des Solutions (Stage d'optimisation)

Pour évaluer la convergence de CEDA au cours de la première étape, nous exécutons une série de tests sur les problèmes de référence bien connus (ZDT1, ...) et nous mesurons l'indicateur hypervolume obtenu avec les deux variantes de la CEDA (CEDA-NSGA2 et CEDA-SPEA2) que ainsi que ceux obtenus avec NSGA2 et SPEA2.

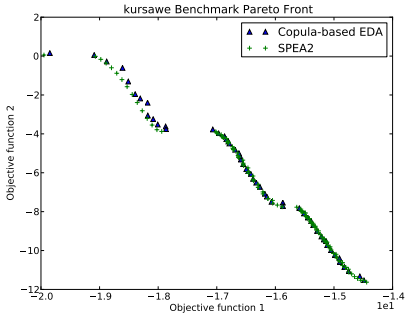
	ZDT1	ZDT2	ZDT3	ZDT4
CEDA on SPEA2	119,932	119,509	127,434	119.82
CEDA on NSGA2	120,178	120,309	126,591	119.3

TABLE 6.1: Moyenne de la métrique hypervolume dans 30 exécutions pour les différents algorithmes sur les benchmarks utilisés.

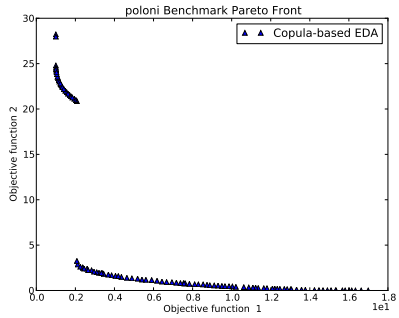
Nous rapportons les résultats de nos tests dans le tableau 6.1. Nous montrons que les deux variantes de CEDA convergent au front Pareto optimal d'une manière qui est similaire à NSGA2 et SPEA2. Cela montre que l'estimateur copule est très bon et com-



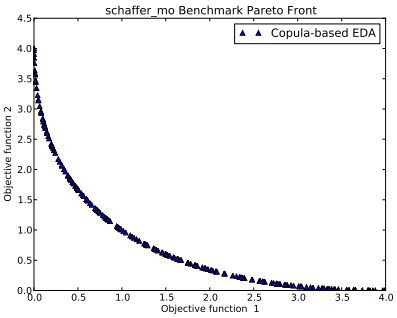
(a) fonseca



(b) Kursawe

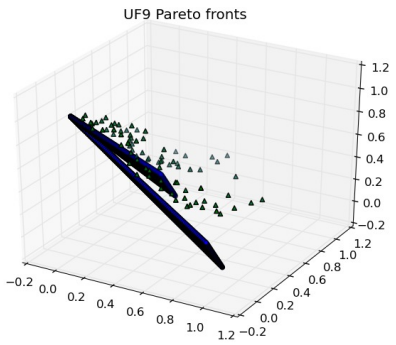


(c) Poloni

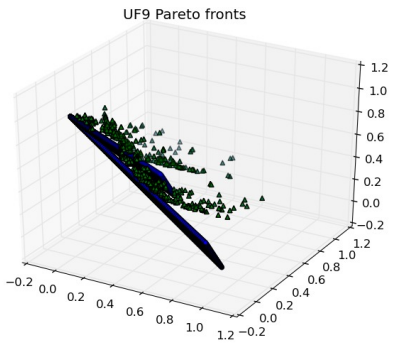


(d) Schaffer

FIGURE 6.3: Front de Pareto des benchmarks Schaffer, Fonseca, Poloni, Kursawe en utilisant CEDA-NSGA2, CEDA-SPEA2 comme méthode de sélection pour CEDA



(a) UF9

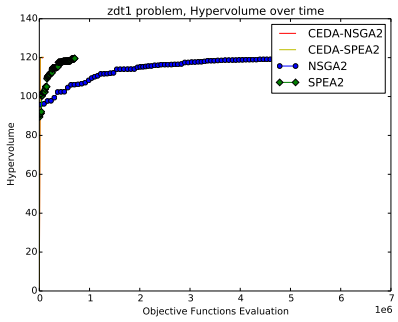


(b) UF9 mis à jour

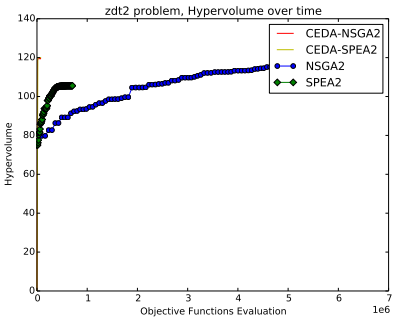
FIGURE 6.4: Front de Pareto des benchmarks UF9 en utilisant CEDA-SPEA2 comme méthode de sélection pour CEDA

parable aux opérateurs génétiques classiques (mutation et de croisement) en termes de reproduction.

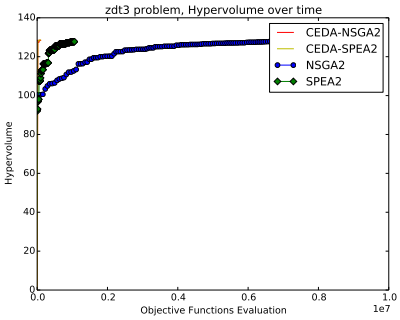
6.3.3 La vitesse de mise à jour des solutions



(a) ZDT1



(b) ZDT2



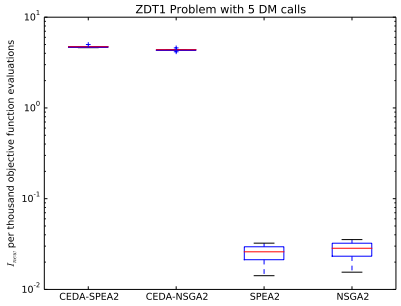
(c) ZDT3

FIGURE 6.5: Hypervolume de Front Pareto des benchmarks ZDT1, ZD2, ZDT3, en utilisant CEDA-NSGA2, CEDA-SPEA2 comme méthode de sélection pour CEDA comparé avec NSGA2 and SPEA2

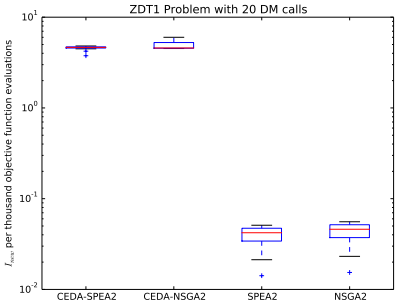
Figure 6.5 montre l'hypervolume du front de Pareto des problèmes de référence ZDT1, ZD2, ZDT3, en utilisant la CEDA-NSGA2, CEDA-SPEA2 comme méthodes de sélection de CEDA comparativement à l'utilisation NSGA2 et SPEA2 seulement. Nous montrons que notre proposition de CEDA génère de nouvelles mises à jour pour les problèmes de référence prise en compte dans un temps considérablement plus petit (mesurée en termes de nombre d'évaluations de la fonction objectif) par rapport aux autres algorithmes. Le calcul du temps commence à partir du début de trouver/générer de nouvelles solutions jusqu'à leur convergence vers le Front Pareto optimal comme illustré dans la Figure 6.2.

6.3.4 Nombres de nouvelles solutions

Dans les figures 6.6, 6.7, 6.8, nous traçons le nombre de nouvelles solutions obtenues lorsqu'un décideur veut générer de nouveaux solutions. Le décideur fait 5 appels

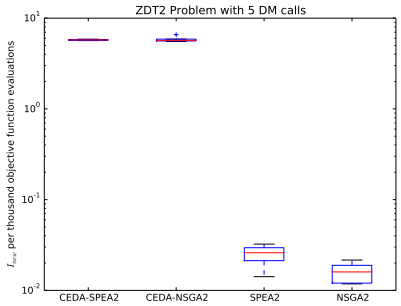


(a) ZDT1 with 5 decision making calls

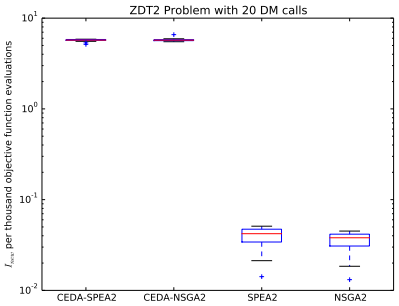


(b) ZDT1 with 20 decision making calls

FIGURE 6.6: Le nombre des nouveaux solutions obtenues (I_{new}) par 1000 évaluations de fonction objectif avec 5 et 20 appels de décideur.

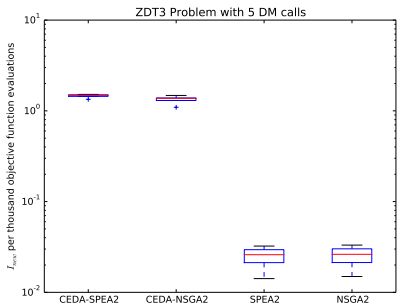


(a) ZDT2 With 5 decision making calls

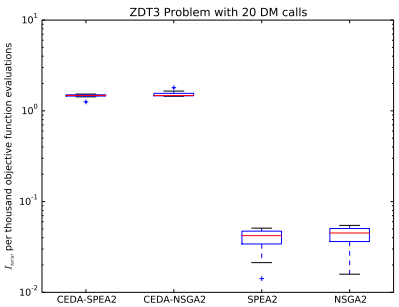


(b) ZDT2 with 20 decision making calls

FIGURE 6.7: Le nombre des nouveaux solutions obtenues (I_{new}) par 1000 évaluations de fonction objectif avec 5 et 20 appels de décideur.



(a) ZDT3 With 5 decision making calls



(b) ZDT3 with 20 decision making calls

FIGURE 6.8: Le nombre des nouveaux solutions obtenues (I_{new}) par 1000 évaluations de fonction objectif avec 5 et 20 appels de décideur.

dans le premier cas et 20 appels dans les secondes pour refléter une prise de décision à divers besoins qui pourraient être satisfaits après différentes nombre d'appels de l'algorithme. Nous traçons la valeur moyenne du nombre de nouvelles solutions en moyenne sur **20** simulations, ainsi que l'écart-type, valeurs minimales et maximales. Nous montrons que les algorithmes basés sur CEDA, tant CEDA-SPEA2 et CEDA-NSGA2, génèrent un nombre significativement plus élevé de nouvelles solutions par évaluation de fonction objectif (de nouvelles solutions sont obtenues avec moins d'évaluations de fonction objectif), par rapport à SPEA2 traditionnelle et algorithmes NSGA-II. Ceci est parce que les techniques basées Copules réduire l'espace de recherche qui se rapproche de front Pareto optimal ce qui rend plus facile de trouver de nouvelles solutions avec un plus petit nombre d'évaluations de fonction objectif par rapport à SPEA2 et NSGA2.

6.3.5 Diversité des solutions

Le tableau 6.2 montre la diversité des résultats obtenus par les algorithmes basés sur CEDA, CEDA-SPEA2 et CEDA-NSGA2, et ceux obtenus avec SPEA2 et l'algorithme NSGA2 traditionnelle. Les valeurs de la diversité obtenues avec des algorithmes basés sur CEDA sont concurrentiels à ceux de SPEA2 et NSGA2. Par exemple, l'algorithme CEDA obtenient de meilleurs diversités sur ZDT1, ZDT3. Sur ZDT2 et ZDT4, CEDA montre une diversité légèrement inférieure. Notez qu'il existe des différences entre la diversité (Δ) de solutions obtenues par les deux variantes de l'algorithme CEDA (CEDA-SPEA2 et CEDA-NSGA2). Cela est dû à la différence de la méthode de sélection de la solution entre SPEA2 et NSGA2.

Les résultats obtenus avec la métrique de diversité montre que notre méthode CEDA proposée trouve toujours un front Pareto optimal réparti sur tout l'espace indépendamment du type du problème des solutions : convexe (par exemple ZDT1) ou non-convexe (par exemple ZDT2), continue (par exemple, ZDT1) ou discontinue (par exemple ZDT3).

	ZDT1	ZDT2	ZDT3	ZDT4	Kur	SCH	FON	POL
SPEA2	0.784	0.755	0.672	0.798	0.852	1.021	0.792	0.972
NSGA2	0.390	0.430	0.738	0.702	0.411	0.477	0.378	0.45
CEDA on SPEA2	0,598	0,603	0,631	0.772	0.611	0.551	0.639	0.72
CEDA on NSGA2	0,592	0,648	0,627	0.789	0.628	0.605	0.645	0.76

TABLE 6.2: Moyen de métrique de diversité Δ dans **30** exécutions pour les différents algorithmes sur les différents benchmarks utilisés.

Conclusion

Nous avons présenté dans ce chapitre les différents benchmarks et les métriques de qualités utilisées dans nos simulations. En outres, nous avons présenté quelques résultats des simulations tests de CEDA sur un ensemble de problèmes de référence traditionnellement utilisées par la communauté pour l'évaluation des algorithmes de résolution de problèmes multiobjectifs et nous avons montré que notre proposition fournit des solutions avec une bonne convergence et diversité par rapport aux algorithmes de l'état de l'art tels que SPEA2 et NSGA2 . Nous avons aussi particulièrement montré que le temps nécessaire pour générer ces solutions est considérablement inférieur à celui requis par des algorithmes de l'état de l'art , ce qui rend notre algorithme approprié pour la génération rapide des solutions de rechange. Les résultats pour les autres benchmarks sont présentés dans les annexes A et B.

Conclusions et perspectives

6.4 Conclusion

On a traité dans cette thèse l'optimisation d'une classe de problèmes avec plusieurs objectifs connus sous le nom des problèmes multiobjectifs. Les définitions des concepts liés aux problèmes multiobjectifs ont été présentées et expliquées dans le chapitre 1. Un problème d'optimisation multiobjectif (MOP) est un problème d'optimisation qui implique des multiples fonctions d'objectifs qui doivent être optimiser simultanément. Ces objectifs sont généralement contradictoires, améliorer un objectif peut dégrader beaucoup d'autres. Dans ces conditions, il n'existe pas une solution unique qui permet d'optimiser toutes les fonctions. Au lieu de cela, il y a généralement un certain nombre de solutions optimales, appelées solutions Pareto, qui sont toutes considérées comme bonnes.

Dans le chapitre 2, nous avons bien détaillé les méthodes d'optimisation multiobjectifs qui utilisent les algorithmes évolutionnaires. Ce type de méthodes utilisent des différentes types d'évaluations et sélections, chacune à ses avantages et ses inconvénients. En outre, dans ce chapitre nous avons expliqué les algorithmes les plus connus dans le domaine d'optimisation multiobjectif par exemple les algorithmes : NSGA-II, SPEA2 et MOEA/D. Les différentes classes de framework aussi ont été décrites dans ce chapitre citons : les frameworks basés dominances, les frameworks basés décompositions et les frameworks basés préférences. Dans cette thèse, nous avons utilisés les deux premiers framework avec une représentation explicite, car on a supposé que le DM n'a pas des connaissances préalables du problème alors il ne peut pas donnée ses préférences.

Les représentations explicites des solutions sont utilisées par une classe des algorithmes évolutionnaires nommée EDA (Estimation of distribution Algorithm). Les Algorithmes à estimation de distribution (EDA) ont été détaillés dans le chapitre 3. Les méthodes de représentations utilisées dans les EDAs ; à longueur fixe, à vecteur de valeur réelle et EDA pour les problèmes de permutation, ont été bien expliquées dans le chapitre 3.

Bien qu'il existe de nombreuses variantes des algorithmes à estimation de distribution, nous avons basé notre travail (CEDA) sur les Copules. Dans la dernière partie, on a commencé par des rappels sur les notions utilisées dans la théorie des copules. Par la suite, nous avons expliqué notre première contribution dans le chapitre 4. Nous avons divisé notre première contribution en deux : la première concerne l'utilisation de CEDA avec les deux variantes de sélection NSGA-II et SPEA2 pour l'optimisation des MOPs (MultiObjective Problem), la deuxième explique comment notre algorithme réagit si le décideur n'est pas satisfait par l'ensemble des solutions optimales trouvé.

Dans le chapitre qui suit (chapitre 5), nous avons présenté notre deuxième contribution, qui est un algorithme CEDA hybride avec un algorithme d'apprentissage automatique (SVM). L'algorithme proposé vise à mettre à jour les solutions trouvées dans la phase d'optimisation sans avoir relancer l'optimisation. On a testé l'algorithme CEDA-SVM proposé sur des benchmarks avec plusieurs objectifs (plus de trois objectifs), qui sont appelés dans la littérature des problèmes "Many-objective".

Les benchmarks utilisés et les métriques de qualités utilisées dans les tests des deux contributions ont été présenté en détail dans le chapitre 6. Les résultats des simulations des algorithmes sont présentés dans les Annexes A et B. Ces résultats montrent que nos deux propositions donnent des meilleurs résultats surtout lorsque on veut avoir des solutions optimales alternatives dans un temps de calcul petit. Néanmoins, si les variables de problèmes sont totalement indépendantes, notre algorithme risque de ne pas trouver des solutions dans la phase d'optimisation ou dans la phase de mise à jour des solutions optimales.

6.5 Perspectives

L'investigation des différentes méthodes de résolutions des problèmes multiobjectifs et many-objective nous a donnée une bonne connaissance sur les points forts de chaque méthode et les points faibles. Aussi nous a permis de bien savoir l'adaptation de ces méthodes sur les problèmes de monde réel et comment peuvent être exploiter dans les problèmes du monde réel. Par conséquent, notre première perspective de ce travail est l'application des méthodes étudiées et proposées sur des problèmes réels tels que le problème de trafic routière et urbain et la bonne gestion des plans routiers. Une autre perspective du travail et l'application de l'algorithme proposé dans les problèmes incertains ou dynamique où les variables de problèmes ou bien les objectifs changent au cours du temps en utilisant les méthodes spécialement baser sur la mémoire et utilisé le modèle

estimé comme mémoire.

Un approfondissement dans le domaine des estimateurs est essentiel pour bien bénéficier des outils existants dans les statistiques et les théories des probabilités pour qu'une bonne explication d'aspect évolutif des algorithmes évolutionnaires. Une autre perspective de travail est l'extension de notre proposition sur les problèmes combinatoires où les variables de problème sont des variables discrètes et surtout les problèmes combinatoires multiobjectifs en utilisant des benchmarks comme le problème de sac à dos multiobjectif, le problème VRP ensuite l'application de la proposition sur des problèmes combinatoires dans le monde réel surtout dans notre pays.

Bibliographie

[1] Djeffal Abdelhamid, Babahenini Mohamed Chaouki, and Taleb-Ahmed Abdelmalik, *A new approach to multi-class svm learning based on oc-svm for huge databases*, Informatics Engineering and Information Science (Azizah Abd Manaf, Akram Zeki, Mazdak Zamani, Suriayati Chuprat, and Eyas El-Qawasmeh, eds.), Communications in Computer and Information Science, vol. 252, Springer Berlin Heidelberg, 2011, pp. 677–690 (English).

[2] S Agrawal, B K Panigrahi, and M K Tiwari, *Multiobjective Particle Swarm Algorithm With Fuzzy Clustering for Electrical Power Dispatch*, Evolutionary Computation, IEEE Transactions on **12** (2008), no. 5, 529–541.

[3] Hazem Radwan Ahmed, *An efficient fitness-based stagnation detection method for particle swarm optimization*, Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion - GECCO Comp '14 (2014), 1029–1032.

[4] Chang Wook Ahn, R S Ramakrishna, and David E Goldberg, *Real-Coded Bayesian Optimization Algorithm : Bringing the Strength of BOA into the Continuous World*, Genetic and Evolutionary Computation – GECCO 2004, 2004, pp. 840–851.

[5] Shaukat Ali, Muhammad Zohaib Iqbal, and Andrea Arcuri, *Improved heuristics for solving OCL constraints using search algorithms*, Proceedings of the 2014 conference on Genetic and evolutionary computation - GECCO '14 (2014), 1231–1238.

[6] Carlos E. Andrade, Flávio K. Miyazawa, and Mauricio G.C. Resende, *Evolutionary algorithm for the k-interconnected multi-depot multi-traveling salesmen problem*, Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference - GECCO '13 (2013), 463.

[7] Ruben Armananzas, Yvan Saeys, Inaki Inza, Miguel Garcia-Torres, Concha Bielza, Yves Van De Peer, and Pedro Larranaga, *Peakbin Selection in Mass Spectrome-*

try Data Using a Consensus Approach with Estimation of Distribution Algorithms, IEEE/ACM Transactions on Computational Biology and Bioinformatics **8** (2011), no. 3, 760–774.

[8] Jaume Bacardit, Michael Stout, Jonathan D Hirst, Kumara Sastry, Xavier Llorà, and Natalio Krasnogor, *Automated alphabet reduction method with evolutionary algorithms for protein structure prediction*, 2007.

[9] Johannes Bader and Eckart Zitzler, *HypE : an algorithm for fast hypervolume-based many-objective optimization.*, Evolutionary computation **19** (2011), no. 1, 45–76.

[10] Ignasi Belda, Sergio Madurga, Xavier Llorà, Marc Martinell, Teresa Tarragó, Mireia G. Piqueras, Ernesto Nicolás, and Ernest Giralt, *ENPDA : An evolutionary structure-based de novo peptide design algorithm*, Journal of Computer-Aided Molecular Design **19** (2005), no. 8, 585–601.

[11] Endika Bengoetxea, Pedro Larraaga, Isabelle Bloch, Aymeric Perchant, and Claudia Boeres, *Inexact graph matching by means of estimation of distribution algorithms*, Pattern Recognition **35** (2002), no. 12, 2867–2880.

[12] P Bickel, P Diggle, S Fienberg, U Gather, I Olkin, and S Zeger, *An Introduction to Copulas*, Springer US, 2006.

[13] Zafer Bingul, *Adaptive genetic algorithms applied to dynamic multiobjective problems*, Applied Soft Computing Journal **7** (2007), no. 3, 791–799.

[14] Sd Bolboaca and Lorentz Jäntschi, *Pearson versus Spearman, Kendall's tau correlation analysis on structure-activity relationships of biologic active compounds*, Leonardo Journal of Sciences (2006), no. 9, 179–200.

[15] _____, *Pearson versus Spearman, Kendall's tau correlation analysis on structure-activity relationships of biologic active compounds*, Leonardo Journal of Sciences (2006), no. 9, 179–200.

[16] Peter A N Bosman, *On empirical memory design, faster selection of bayesian factorizations and parameter-free gaussian edas*, Belgian/Netherlands Artificial Intelligence Conference, 2009, pp. 285–286.

[17] Peter A N Bosman and Edwin D de Jong, *Learning Probabilistic Tree Grammars for Genetic Programming*, Parallel Problem Solving from Nature - PPSN VIII, vol. 3242, 2004, pp. 192–201.

[18] Peter A. N. Bosman and Dirk Thierens, *Continuous Iterated Density Estimation Evolutionary Algorithms Within the IDEA Framework*, Optimization by Building and Using Probabilistic Models OBUPM Workshop at the Genetic and Evolutionary Computation Conference - GECCO-2000 (2000), 197–200.

[19] Peter A N Bosman and Dirk Thierens, *Multi-objective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms*, International Journal of Approximate Reasoning **31** (2002), no. 3, 259–289.

[20] Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Slowinski, *Multiobjective optimization : interactive and evolutionary approaches*, vol. 5252, 2008.

[21] D Brockhoff and E Zitzler, *Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods*, {IEEE} Congress on Evolutionary Computation, 2007. {CEC} 2007, 2007, pp. 2086–2093.

[22] Dimo Brockhoff and Eckart Zitzler, *Objective reduction in evolutionary multiobjective optimization : theory and applications.*, Evolutionary computation **17** (2009), no. 2, 135–166.

[23] Rainer E. Burkard, Eranda Cela, Panos M. Pardalos, and Leonidas S. Pitsoulis, *The Quadratic Assignment Problem*, Handbook of Combinatorial Optimization (1998), 1–71.

[24] Z Cai, Z Cai, and Y Wang, *A Multiobjective Optimization-Based Evolutionary Algorithm for Constrained Optimization*, Evolutionary Computation, IEEE Transactions on **10** (2006), no. 6, 658–675.

[25] V. Černý, *Thermodynamical approach to the traveling salesman problem : An efficient simulation algorithm*, Journal of Optimization Theory and Applications **45** (1985), no. 1, 41–51.

[26] Shelvin Chand and Markus Wagner, *Evolutionary many-objective optimization : A quick-start guide*, Surveys in Operations Research and Management Science **20** (2015), no. 2, 35–42.

[27] Abraham Charnes and William Wager Cooper, *Goal programming and multiple objective optimizations : Part 1*, European Journal of Operational Research **1** (1977), no. 1, 39–54.

[28] Ying-ping Chen and Chao-Hong Chen, *Enabling the extended compact genetic algorithm for real-parameter optimization by using adaptive discretization.*, Evolutionary computation **18** (2010), no. 2, 199–228.

[29] Abdelhakim Cheriet and Foudil Cherif, *A Posteriori Pareto Front Diversification Using a Copula-Based Estimation of Distribution Algorithm*, International Journal of Advanced Computer Science and Applications **6** (2015), no. 12 (en).

[30] Umberto Cherubini, Elisa Luciano, and Walter Vecchiato, *Copula Methods in Finance*, 2013.

[31] Carlos a Coello Coello, Gary B Lamont, and David a Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems Second Edition*, 2007.

[32] Carlos a. Coello Coello, *Theoretical and numerical constraint-handling techniques used with evolutionary algorithms : A survey of the state of the art*, Computer Methods in Applied Mechanics and Engineering **191** (2002), no. 11-12, 1245–1287.

[33] Carlos A. Coello Coello and Margarita Reyes-Sierra, *Multi-Objective Particle Swarm Optimizers : A Survey of the State-of-the-Art*, 2006.

[34] Jeremy S. De Bonet, Charles L. Isbell, and Paul Viola, *MIMIC : Finding Optima by Estimating Probability Densities*, Advances in Neural Information Processing Systems **9** (1997), 424.

[35] K. Deb and H. Jain, *An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i : Solving problems with box constraints*, Evolutionary Computation, IEEE Transactions on **18** (2014), no. 4, 577–601.

[36] K. Deb, A Pratap, S. Agarwal, and T. Meyarivan, *A fast and elitist multiobjective genetic algorithm : Nsga-ii*, Evolutionary Computation, IEEE Transactions on **6** (2002), no. 2, 182–197.

[37] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan, *A fast and elitist multiobjective genetic algorithm : NSGA-II*, IEEE Transactions on Evolutionary Computation **6** (2002), no. 2, 182–197.

[38] Kalyanmoy Deb and Amit Saha, *Finding multiple solutions for multimodal optimization problems using a multi-objective evolutionary approach*, Proceedings of the

12th annual conference on Genetic and evolutionary computation - GECCO '10, 2010, p. 447.

[39] Kalyanmoy Deb, Lothar Thiele, and Marco Laumanns, *Scalable test problems for evolutionary multiobjective optimization*, Evolutionary Multiobjective (2005), no. 1990, 1–27.

[40] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler, *Scalable multi-objective optimization test problems*, Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002 **1** (2002), 825–830.

[41] Karl Doerner, Walter J. Gutjahr, Richard F. Hartl, Christine Strauss, and Christian Stummer, *Pareto ant colony optimization : A metaheuristic approach to multiobjective portfolio selection*, Annals of Operations Research **131** (2004), no. 1-4, 79–99.

[42] Weishan Dong and Xin Yao, *Unified eigen analysis on multivariate Gaussian based estimation of distribution algorithms*, Information Sciences **178** (2008), no. 15, 3000–3023.

[43] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Natural Computing Series, vol. 1, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[44] Ahmed Elhossini, Shawki Areibi, and Robert Dony, *Strength Pareto particle swarm optimization and hybrid EA-PSO for multi-objective optimization.*, Evolutionary computation **18** (2010), no. 1, 127–56.

[45] Paul Embrechts, Alexander McNeil, and Daniel Straumann, *CORRELATION AND DEPENDENCE IN RISK MANAGEMENT : PROPERTIES AND PITFALLS*, 2002.

[46] Paul Embrechts, Er Mcneil, and Daniel Straumann, *Correlation : pitfalls and alternatives*, Risk Magazine, Citeseer, 1999.

[47] Marco Farina, Kalyanmoy Deb, and Paolo Amato, *Dynamic multiobjective optimization problems : Test cases, approximations, and applications*, IEEE Transactions on Evolutionary Computation **8** (2004), no. 5, 425–442.

[48] Peter J. Fleming, Robin C. Purshouse, and Robert J. Lygoe, *Many-Objective Optimization : An Engineering Design Perspective*, Evolutionary Multi-Criterion Optimization (Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, eds.), Lecture Notes in Computer Science, Springer Berlin Heidelberg, mar 2005, pp. 14–32 (en).

[49] Ying Gao, Lingxi Peng, Fufang Li, Miao Liu, and Xiao Hu, *Pareto-based multi-objective estimation of distribution algorithm with gaussian copulas and application in RFID network planning*, Advanced Computational Intelligence (ICACI), 2012 IEEE Fifth International Conference on, 2012, pp. 370–373.

[50] C. García-Martínez, O. Cordón, and F. Herrera, *A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP*, European Journal of Operational Research **180** (2007), no. 1, 116–148.

[51] Mario Garza-Fabre, Gregorio Toscano Pulido, and Carlos a Coello Coello, *Ranking methods for many-objective optimization*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Arturo Hernández Aguirre, Raúl Monroy Borja, and Carlos Alberto Reyes Garciá, eds.), Lecture Notes in Computer Science, vol. 5845 LNAI, Springer Berlin Heidelberg, 2009, pp. 633–645 (en).

[52] Christian Genest and Louis-Paul Rivest, *Statistical Inference Procedures for Bivariate Archimedean Copulas*, 1993, pp. 1034–1043.

[53] Chi Keong Goh and Key Chen Tan, *A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization*, IEEE Transactions on Evolutionary Computation **13** (2009), no. 1, 103–127.

[54] D Goldberg, *Real-coded genetic algorithms, virtual alphabets, and blocking*, Complex Systems (1991), 139–167.

[55] David E Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, vol. Addison-We, 1989.

[56] David E. Goldberg, Kumara Sastry, and Xavier Llorà, *Toward routine billion-variable optimization using genetic algorithms*, Complexity **12** (2007), no. 3, 27–29.

[57] Maoguo Gong, Licheng Jiao, Haifeng Du, and Liefeng Bo, *Multiobjective immune algorithm with nondominated neighbor-based selection.*, Evolutionary computation **16** (2008), no. 2, 225–255.

[58] Wenyin Gong and Zhihua Cai, *An improved multiobjective differential evolution based on Pareto-adaptive ϵ{lunate}-dominance and orthogonal design*, European Journal of Operational Research **198** (2009), no. 2, 576–601.

[59] David Hadka and Patrick Reed, *Diagnostic Assessment of Search Controls and Failure Modes in Many-Objective Evolutionary Optimization*, Evolutionary Computation **20** (2012), no. 3, 423–452.

[60] Georges R. Harik, Fernando G. Lobo, and David E. Goldberg, *The compact genetic algorithm*, IEEE Transactions on Evolutionary Computation **3** (1999), no. 4, 287–297.

[61] GR Harik, *Finding multimodal solutions using restricted tournament selection*, Proceedings of the Sixth International Conference on Genetic Algorithms, 1995, pp. 24–31.

[62] Yoshihiko Hasegawa and Hitoshi Iba, *Estimation of Distribution Algorithm Based on Probabilistic Grammar with Latent Annotations*, 2007 IEEE Congress on Evolutionary Computation (2007), 1043–1050.

[63] Mark Hauschild and Martin Pelikan, *Intelligent Bias of Network Structures in the Hierarchical BOA*, Genetic and Evolutionary Computation Conference, no. 2009005, 2009, pp. 413–420.

[64] ———, *An introduction and survey of estimation of distribution algorithms*, Swarm and Evolutionary Computation **1** (2011), no. 3, 111–128.

[65] ———, *An introduction and survey of estimation of distribution algorithms*, Swarm and Evolutionary Computation **1** (2011), no. 3, 111–128.

[66] John H Holland, *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence.*, U Michigan Press, 1975.

[67] Zhi Hua Hu, *A multiobjective immune algorithm based on a multiple-affinity model*, European Journal of Operational Research **202** (2010), no. 1, 60–72.

[68] V. L. Huang, S. Z. Zhao, R. Mallipeddi, and P. N. Suganthan, *Multi-objective optimization using self-adaptive differential evolution algorithm*, 2009 IEEE Congress on Evolutionary Computation (2009), 190–194.

[69] S. Huband, P. Hingston, L. While, and L. Barone, *An evolution strategy with probabilistic mutation for multi-objective optimisation*, 2003 Congress on Evolutionary Computation, CEC 2003 - Proceedings, vol. 4, 2003, pp. 2284–2291.

[70] Christian Igel, Nikolaus Hansen, and Stefan Roth, *Covariance matrix adaptation for multi-objective optimization.*, Evolutionary computation **15** (2007), no. 1, 1–28.

[71] A. Inselberg and B. Dimsdale, *Parallel coordinates : a tool for visualizing multi-dimensional geometry*, Proceedings of the First IEEE Conference on Visualization : Visualization ‘90, IEEE Comput. Soc. Press, oct 1990, pp. 361–378.

[72] H. Ishibuchi and T. Murata, *A multi-objective genetic local search algorithm and its application to flowshop scheduling*, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **28** (1998), no. 3, 392–403.

[73] Himanshu Jain and Kalyanmoy Deb, *An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, Part II : Handling constraints and extending to an adaptive approach*, IEEE Transactions on Evolutionary Computation **18** (2014), no. 4, 602–622.

[74] A. Jaszkiewicz, *On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - a comparative experiment*, IEEE Transactions on Evolutionary Computation **6** (2002), no. 4, 402–412.

[75] Piotr Jaworski, Fabrizio Durante, Wolfgang Karl Hardle, and Tomasz Rychlik, *Copula theory and its applications*, Springer, 2010.

[76] I. Y. Kim and O. L. De Weck, *Adaptive weighted-sum method for bi-objective optimization : Pareto front generation*, Structural and Multidisciplinary Optimization **29** (2005), no. 2, 149–158.

[77] S Kirkpatrick, C D Gelatt, and M P Vecchi, *Optimization by Simulated Annealing*, Science **220** (1983), no. 4598, pp. 671–680.

[78] J D Knowles and D W Corne, *Approximating the nondominated front using the Pareto Archived Evolution Strategy.*, Evolutionary computation **8** (2000), no. 2, 149–172.

[79] J.B. Kollat and P.M. Reed, *Comparing state-of-the-art evolutionary multi-objective algorithms for long-term groundwater monitoring design*, Advances in Water Resources **29** (2006), no. 6, 792–807.

[80] Adriana Lara, Gustavo Sanchez, C.a. Coello Coello, and O. Schutze, *HCS : A New Local Search Strategy for Memetic Multiobjective Evolutionary Algorithms*, IEEE Transactions on Evolutionary Computation **14** (2010), no. 1, 112–132.

[81] P Larrañaga, R Etxeberria, J a Lozano, and J M Peña, *Optimization in Continuous Domains by Learning and Simulation of Gaussian Networks*, Proceedings of the 2th Genetic and Evolutionary Computation Conference, GECCO 2000 (2000), 201–204.

[82] P Larrañaga and J A Lozano, *Estimation of Distribution Algorithms : A New Tool for Evolutionary Computation*, vol. 2, 2002.

[83] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler, *Combining convergence and diversity in evolutionary multiobjective optimization.*, Evolutionary computation **10** (2002), no. 3, 263–282.

[84] Marco Laumanns, Lothar Thiele, and Eckart Zitzler, *An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method*, European Journal of Operational Research **169** (2006), no. 3, 932–942.

[85] Wen Fung Leong and Gary G. Yen, *PSO-based multiobjective optimization with dynamic population size and adaptive local archives*, IEEE Transactions on Systems, Man, and Cybernetics, Part B : Cybernetics **38** (2008), no. 5, 1270–1293.

[86] Bin-Bin Li and Ling Wang, *A hybrid quantum-inspired genetic algorithm for multiobjective flow shop scheduling.*, IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society **37** (2007), no. 3, 576–591.

[87] Bingdong Li, Jinlong Li, Ke Tang, and Xin Yao, *Many-Objective Evolutionary Algorithms*, ACM Computing Surveys **48** (2015), no. 1, 1–35.

[88] Hui Li, Qingfu Zhang, Edward Tsang, and John A. Ford, *Hybrid Estimation of Distribution Algorithm for Multiobjective Knapsack Problem*, Lecture Notes in Computer Science, Springer Berlin Heidelberg, apr 2004, pp. 145–154 (en).

[89] Hui Li Hui Li and Qingfu Zhang Qingfu Zhang, *Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II*, IEEE Transactions on Evolutionary Computation **13** (2009), no. 2, 284–302.

[90] J Li and U Aickelin, *A Bayesian optimization algorithm for the nurse scheduling problem*, 2003 Congress on Evolutionary Computation, CEC 2003, vol. 3, 2003, pp. 2149–2156.

[91] Jingpeng Li and Uwe Aickelin, *Scalable Optimization via Probabilistic Modeling*, vol. 33, 2006.

[92] D.S. Liu, K.C. Tan, S.Y. Huang, C.K. Goh, and W.K. Ho, *On solving multiobjective bin packing problems using evolutionary particle swarm optimization*, European Journal of Operational Research **190** (2008), no. 2, 357–382.

[93] Fernando G. Lobo, *Lost gems of EC*, ACM SIGEVOlution **2** (2007), no. 2, 14–15.

[94] Eliane Maria Loiola, Nair Maria Maia de Abreu, Paulo Oswaldo Boaventura-Netto, Peter Hahn, and Tania Querido, *A survey for the quadratic assignment problem*, European Journal of Operational Research **176** (2007), no. 2, 657–690.

[95] Moshe Looks, *Levels of abstraction in modeling and sampling*, Proceedings of the 8th annual conference on Genetic and evolutionary computation - GECCO '06 (New York, New York, USA), GECCO '06, ACM Press, 2006, p. 429.

[96] Robert I. McKay, Nguyen Xuan Hoai, Peter Alexander Whigham, Yin Shan, and Michael O'neill, *Grammar-based Genetic programming : A survey*, Genetic Programming and Evolvable Machines **11** (2010), no. 3-4, 365–396.

[97] Teresa Miquélez, Endika Bengoetxea, Pedro Larrañaga, Jose Antonio Lozano, Iñaki Inza, Julio Madera, Enrique Alba, Alberto Ochoa, Victor Robles, Jose Peña, María Pérez, Vanessa Herves, Yvan Saeys, Sven Degroeve, Yves Van de Peer, M. Flores, José Gámez, José Puerta, Jiri Ocenasek, Marta Soto, and Nikolaus Hansen, *Towards a New Evolutionary Computation*, Towards a New Evolutionary Computation **192** (2006), no. 2006, 39–50.

[98] Heinz Muhlenbein and Thilo Mahnig, *The factorized distribution algorithm for additively decomposed functions*, Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999, vol. 1, 1999, pp. 752–759.

[99] Heinz Mühlenbein and G Paaß, *From Recombination of Genes to the Estimation of Distributions I. Binary Parameters*, Parallel Problem Solving from Nature — PPSN IV, vol. 1141, 1996, pp. 178–187.

[100] T Murata and H Ishibuchi, *MOGA : multi-objective genetic algorithms*, Evolutionary Computation, 1995., IEEE International Conference on **1** (1995), 289.

[101] Roger B Nelsen, *An introduction to copulas*, Springer, 2006.

[102] J Ocenasek, S Kern, N Hansen, and P Koumoutsakos, *A Mixed Bayesian Optimization Algorithm with variance adaptation*, Parallel Problem Solving From Nature – PPSN VIII **3242** (2004), 352–361.

[103] T. Okabe, Y. Jin, B. Sendoff, and M. Olhofer, *Voronoi-based estimation of distribution algorithm for multi-objective optimization*, Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753) **2** (2004), 1594–1601.

[104] L Paquete and T Stutzle, *A two-phase local search for the biobjective traveling salesman problem*, Evolutionary Multi-Criterion Optimization, Proceedings **2632** (2003), 479–493.

[105] M. Pelikan, D.E. Goldberg, and S. Tsutsui, *Hierarchical Bayesian optimization algorithm : toward a new generation of evolutionary algorithms*, vol. 3, 2003.

[106] Martin Pelikan and David E. Goldberg, *Escaping Hierarchical Traps with Competent Genetic Algorithms*, Proceedings of the Genetic and Evolutionary Computation Conference (Gecco'01), Morgan Kaufmann, 2001, pp. 511–518.

[107] ———, *Hierarchical Bayesian Optimization Algorithm*, Scalable Optimization via Probabilistic Modeling From Algorithms to Applications (Dr Martin Pelikan, Kumara Sastry, and Dr Erick CantúPaz, eds.), Studies in Computational Intelligence, Springer Berlin Heidelberg, 2006, pp. 63–90 (en).

[108] Martin Pelikan, David E Goldberg, and Erick Cantú-Paz, *BOA : The Bayesian Optimization Algorithm*, Genetic and Evolutionary Computation **1** (1999), 525–532.

[109] Martin Pelikan, David E. Goldberg, and Erick Cantú-Paz, *Linkage problem, distribution estimation, and Bayesian networks.*, Evolutionary computation **8** (2000), no. 3, 311–40.

[110] Martin Pelikan, David E. Goldberg, and Shigeyoshi Tsutsui, *Getting the best of both worlds : Discrete and continuous genetic and evolutionary algorithms in concert*, Information Sciences **156** (2003), no. 3-4, 147–171.

[111] Martin Pelikan, Kumara Sastry, and David E. Goldberg, *Multiobjective hBOA, clustering, and scalability*, Proceedings of the 2005 conference on Genetic and evolutionary computation - GECCO '05 (2005), 663.

[112] Bo-yang Qu and Ponnuthurai-Nagaratnam Suganthan, *Multi-objective differential evolution with diversity enhancement*, Journal of Zhejiang University {SCIENCE} C **11** (2010), no. 7, 538–543.

[113] Stefan Rudlof and Mario Köppen, *Stochastic Hill Climbing with Learning by Vectors of Normal Distribution*, WSC1, 1996, pp. 60–70.

[114] Abdellah Salhi, José Antonio Vázquez Rodríguez, and Qingfu Zhang, *An estimation of distribution algorithm with guided mutation for a complex flow shop scheduling problem*, Proceedings of the 9th annual conference on Genetic and evolutionary computation - GECCO '07, 2007, p. 570.

[115] Rogelio Salinas-Gutiérrez, Arturo Hernández-Aguirre, and Enrique R Villa-Diharce, *Using copulas in estimation of distribution algorithms*, MICAI 2009 : Advances in Artificial Intelligence, Springer, 2009, pp. 658–668.

[116] Rogelio Salinas-Gutiérrez, Arturo Hernández-Aguirre, and Enrique R. Villa-Diharce, *Estimation of distribution algorithms based on copula functions*, Proceedings of the 13th annual conference companion on Genetic and evolutionary computation - GECCO '11 (2011), no. 1, 795.

[117] Rogelio Salinas-Gutiérrez, Arturo Hernández-Aguirre, and Enrique R. Villa-Diharce, *Estimation of distribution algorithms based on copula functions*, Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation (New York, NY, USA), GECCO '11, ACM, 2011, pp. 795–798.

[118] R Santana, P Larranaga, and J A Lozano, *Protein folding in simplified models with estimation of distribution algorithms*, Ieee Transactions on Evolutionary Computation **12** (2008), no. 4, 418–438.

[119] Roberto Santana, *Estimation of distribution algorithms with Kikuchi approximations.*, Evolutionary computation **13** (2005), no. 1, 67–97.

[120] J.D. Schaffer, *Multiple objective optimization with vector evaluated genetic algorithms*, ... the 1st international Conference on Genetic Algorithms (1985), no. JANUARY 1985, 93–100.

[121] J Schwarz and J. Očenášek, *A problem knowledge-based evolutionary algorithm KBOA for hypergraph bisectioning*, Proceedings of the 4th Joint Conference on Knowledge-Based Software Engineering, 2000, pp. 51–58.

[122] Mich Sebag and Antoine Ducoulombier, *Extending population-based incremental learning to continuous search spaces*, Parallel Problem Solving from Nature **5** (1998), 418–427.

[123] Siddhartha Shakya, Alexander Brownlee, John McCall, François Fournier, and Gilbert Owusu, *A fully multivariate DEUM algorithm*, 2009 IEEE Congress on Evolutionary Computation, CEC 2009, 2009, pp. 479–486.

[124] Hanif D Sherali, *Equivalent weights for lexicographic multi-objective programs : characterizations and computations*, European Journal of Operational Research **11** (1982), no. 4, 367–379.

[125] Vui Ann Shim, KC Tan, and CY Cheong, *A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem*, Systems, Man, and Cybernetics, Part C : Applications and Reviews, IEEE Transactions on **42** (2012), no. 5, 682–691.

[126] Bw. Silverman, *Density estimation for statistics and data analysis*, Chapman and Hall **37** (1986), no. 1, 1–22.

[127] Hemant Kumar Singh, Amitay Isaacs, and Tapabrata Ray, *A Pareto corner search evolutionary algorithm and dimensionality reduction in many-objective optimization problems*, IEEE Transactions on Evolutionary Computation **15** (2011), no. 4, 539–556.

[128] Ankur Sinha, Pekka Korhonen, Jyrki Wallenius, and Kalyanmoy Deb, *An interactive evolutionary multi-objective optimization algorithm with a limited number of decision maker calls*, European Journal of Operational Research **233** (2014), no. 3, 674–688.

[129] M Sklar, *Fonctions de répartition à n dimensions et leurs marges*, Université Paris 8, 1959.

[130] N. Srinivas and Kalyanmoy Deb, *Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms*, Evolutionary Computation **2** (1994), no. 3, 221–248.

[131] 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** (2006), no. 2, 463–495.

[132] Raymond Tran, Junhua Wu, Christopher Denison, Thomas Ackling, Markus Wagner, and Frank Neumann, *Fast and effective multi-objective optimisation of wind turbine placement*, Fifteenth annual conference on Genetic and evolutionary computation conference (2013), 1381–1388.

[133] S Tsutsui, *Probabilistic Model-Building Genetic Algorithms in Permutation Representation Domain Using Edge Histogram*, Parallel Problem Solving from Nature — PPSN VII (2002), 224–233.

[134] S. Tsutsui, *Node Histogram vs. Edge Histogram : A Comparison of Probabilistic Model-Building Genetic Algorithms in Permutation Domains*, 2006 IEEE International Conference on Evolutionary Computation (2006), 1939–1946.

[135] S. Tsutsui, M. Pelikan, and D.E. Goldberg, *Evolutionary algorithm using marginal histogram models in continuous domain*, Proceedings of the Genetic and Evolutionary Computation Conference Workshop, 2001, pp. 230–233.

[136] S. Tsutsui, Martin Pelikan, and D.E. Goldberg, *Using edge histogram models to solve permutation problems with probabilistic model-building genetic algorithms*, IlliGAL Report (2003).

[137] D.a. Van Veldhuizen and G.B. Lamont, *On measuring multiobjective evolutionary algorithm performance*, Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512) **1** (2000), 204–211 vol.1.

[138] Andrzej P. Wierzbicki, *The Use of Reference Objectives in Multiobjective Optimization*, Multiple Criteria Decision Making Theory and Application (1980), 468–486.

[139] ———, *A mathematical basis for satisficing decision making*, Mathematical Modelling **3** (1982), no. 5, 391–405.

[140] Y G Woldesenbet, G G Yen, and B G Tessema, *Constraint Handling in Multiobjective Evolutionary Optimization*, Evolutionary Computation, IEEE Transactions on **13** (2009), no. 3, 514–525.

[141] Man-Leung Wong, *Parallel multi-objective evolutionary algorithms on graphics processing units*, GECCO '09 : Proceedings of the 11th annual conference companion on Genetic and evolutionary computation conference, 2009, pp. 2515–2522.

[142] Bing Xue, Mengjie Zhang, Yan Dai, and Will N Browne, *Pso for feature construction and binary classification*, Proceedings of the 15th annual conference on Genetic and evolutionary computation, ACM, 2013, pp. 137–144.

[143] Y. Y. Haimes, L. S. Lasdon and D. A. Wismer, *On a Bicriterion Formulation of the Problems of Integrated System Identification and System Optimization*, IEEE Transactions on Systems, Man, and Cybernetics **1** (1971), no. 3, 296–297.

[144] Chao Yang, Shuming Peng, Bin Jiang, Lei Wang, and Renfa Li, *Hyper-heuristic genetic algorithm for solving frequency assignment problem in TD-SCDMA*, Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion - GECCO Comp '14 (2014), 1231–1238.

[145] Jie Yao, Nawwaf Kharma, and Peter Grogono, *Bi-Objective multipopulation genetic algorithm for multimodal function optimization*, IEEE Transactions on Evolutionary Computation **14** (2010), no. 1, 80–102.

[146] Amelia Zafra, Eva L Gibaja, and Sebastián Ventura, *Multiple instance learning with multiple objective genetic programming for web mining*, Applied Soft Computing **11** (2011), no. 1, 93–102.

[147] Qingfu Zhang and Hui Li, *MOEA/D : A multiobjective evolutionary algorithm based on decomposition*, IEEE Transactions on Evolutionary Computation **11** (2007), no. 6, 712–731.

[148] Qingfu Zhang, Jianyong Sun, and Edward Tsang, *An evolutionary algorithm with guided mutation for the maximum clique problem*, IEEE Transactions on Evolutionary Computation **9** (2005), no. 2, 192–200.

[149] Qingfu Zhang, Aimin Zhou, and Yaochu Jin, *RM-MEDA : A regularity model-based multiobjective estimation of distribution algorithm*, IEEE Transactions on Evolutionary Computation **12** (2008), no. 1, 41–63.

[150] Qingfu Zhang, Aimin Zhou, Shizheng Zhao, Ponnuthurai Nagaratnam Suganthan, Wudong Liu, and Santosh Tiwari, *Multiobjective optimization test instances for the cec 2009 special session and competition*, University of Essex, Colchester, UK and Nanyang technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, technical report **264** (2008).

[151] Yang Zhang and Peter I. Rockett, *A generic optimising feature extraction method using multiobjective genetic programming*, Applied Soft Computing **11** (2011), no. 1, 1087–1097.

[152] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang, *Multiobjective evolutionary algorithms : A survey of the state of the art*, Swarm and Evolutionary Computation **1** (2011), no. 1, 32–49.

[153] ———, *Multiobjective evolutionary algorithms : A survey of the state of the art*, Swarm and Evolutionary Computation **1** (2011), no. 1, 32–49.

[154] E Zitzler, K Deb, and L Thiele, *Comparison of multiobjective evolutionary algorithms : empirical results.*, Evolutionary computation **8** (2000), no. 2, 173–195.

[155] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele, *Comparison of multiobjective evolutionary algorithms : Empirical results*, Evolutionary computation **8** (2000), no. 2, 173–195.

[156] Eckart Zitzler, Marco Laumanns, and Lothar Thiele, *SPEA2 : Improving the Strength Pareto Evolutionary Algorithm*, Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems, 2001, pp. 95–100.

[157] Eckart Zitzler, Marco Laumanns, and Lothar Thiele, *SPEA2 : Improving the strength pareto evolutionary algorithm for multiobjective optimization*, Evolutionary Methods for Design, Optimisation, and Control, CIMNE, Barcelona, Spain, 2002, pp. 95–100.

[158] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert Da Fonseca, *Performance assessment of multiobjective optimizers : An analysis and review*, IEEE Transactions on Evolutionary Computation **7** (2003), no. 2, 117–132.

Annexe A

Les Fronts Pareto des problèmes Many-objective

Dans cette Annexe on va présenter les différents fronts Pareto des problèmes utilisés pour l'optimisation Many-objective. Les problèmes utilisés sont les problèmes DTLZ1, DTLZ2, DTLZ3, DTLZ4 avec des différents dimensions commençant par trois dimensions ensuite 5,10 et 15 dimensions. Chaque problème est optimisé avec l'algorithme MOEA/D et deux différents types de décomposition. Les résultats montrent les fronts de Pareto avant et après l'utilisation de procédure de mise à jour proposée par l'algorithme CEDA-SVM.

A.1 Les résultats pour les problèmes Many-objective utilisés

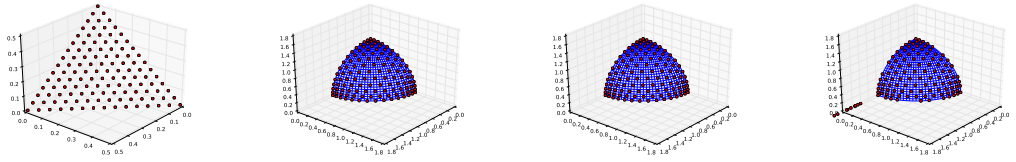


FIGURE A.1: DTLZ(1,2,3,4) benchmarks avec 3 Dimensions utilisant moeadb.

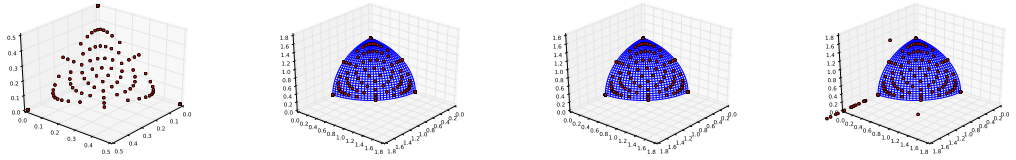


FIGURE A.2: DTLZ(1,2,3,4) benchmarks avec 3 Dimensions utilisant moeadtch.

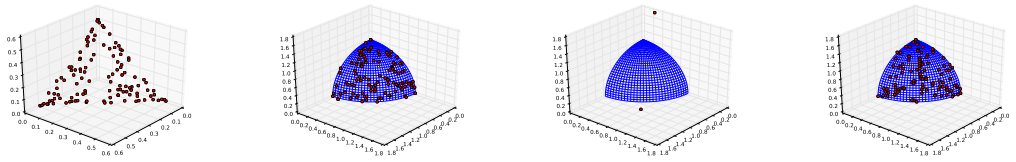


FIGURE A.3: DTLZ(1,2,3,4) benchmarks avec 3 Dimensions utilisant nsga2.

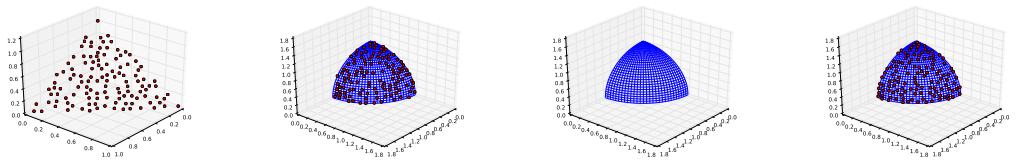


FIGURE A.4: DTLZ(1,2,3,4) benchmarks avec 3 Dimensions utilisant spea2.

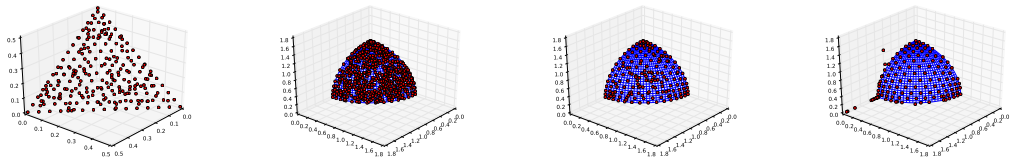


FIGURE A.5: DTLZ(1,2,3,4) benchmarks avec 3 Dimensions utilisant moeadbUpdate.

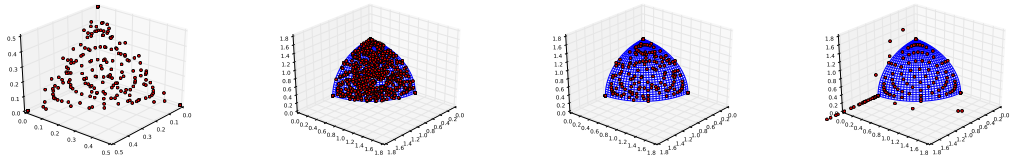


FIGURE A.6: DTLZ(1,2,3,4) benchmarks avec 3 Dimensions utilisant moeadtchUpdate.

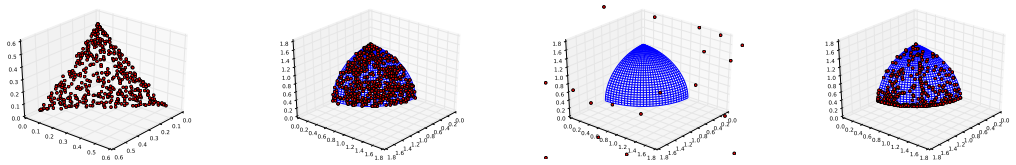


FIGURE A.7: DTLZ(1,2,3,4) benchmarks avec 3 Dimensions utilisant nsga2Update.

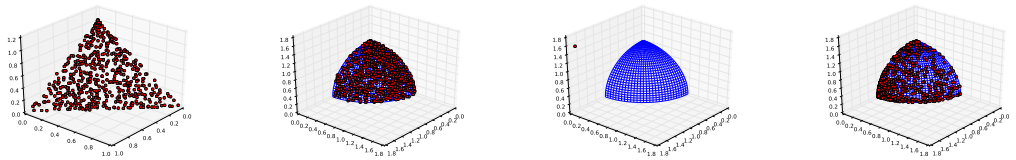


FIGURE A.8: DTLZ(1,2,3,4) benchmarks avec 3 Dimensions utilisant spea2Update.

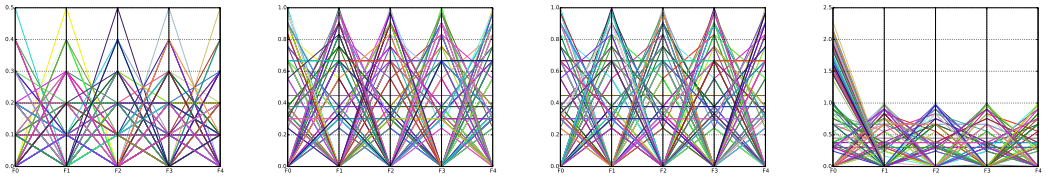


FIGURE A.9: DTLZ(1,2,3,4) benchmarks avec 5 Dimension utilisant MOEA/D-B.

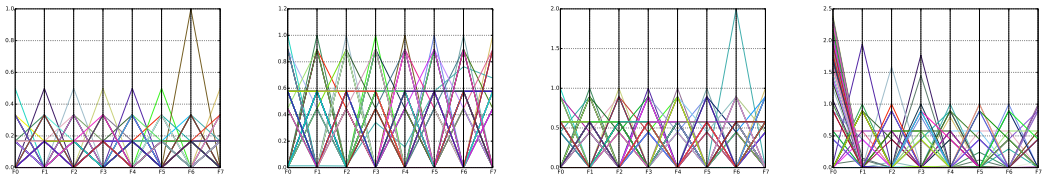


FIGURE A.10: DTLZ(1,2,3,4) benchmarks avec 8 Dimension utilisant MOEA/D-B.

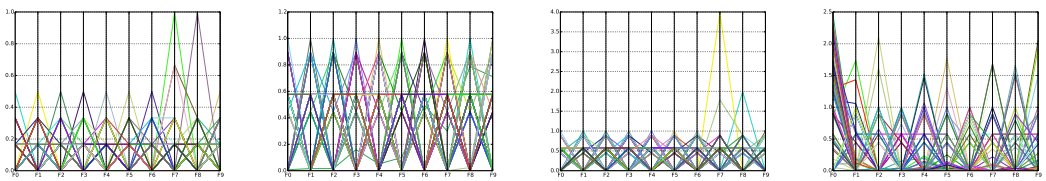


FIGURE A.11: DTLZ(1,2,3,4) benchmarks avec 10 Dimension utilisant MOEA/D-B.

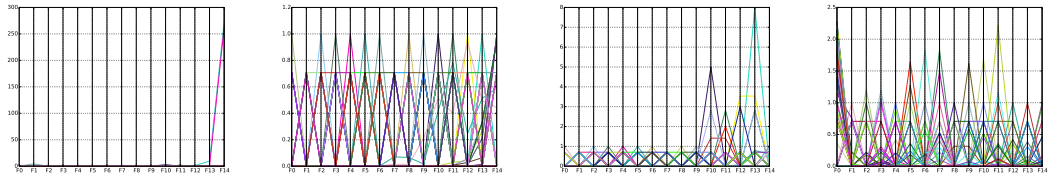


FIGURE A.12: DTLZ(1,2,3,4) benchmarks avec 15 Dimension utilisant MOEA/D-B.

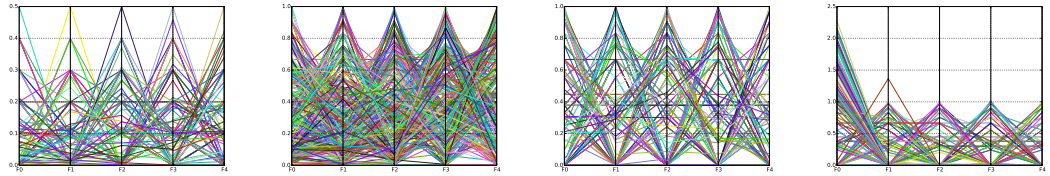


FIGURE A.13: DTLZ(1,2,3,4) benchmarks avec 5 Dimension utilisant U-MOEA/D-B.

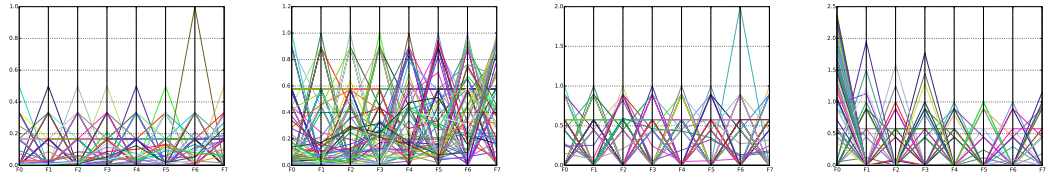


FIGURE A.14: DTLZ(1,2,3,4) benchmarks avec 8 Dimension utilisant U-MOEA/D-B.

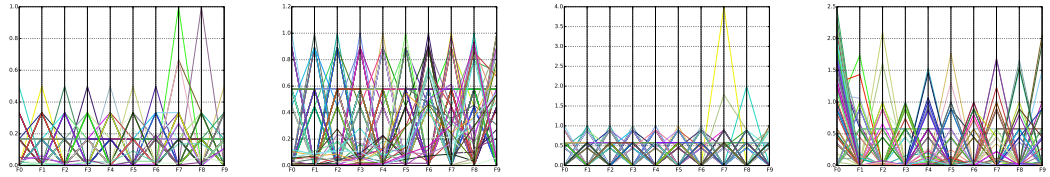


FIGURE A.15: DTLZ(1,2,3,4) benchmarks avec 10 Dimension utilisant U-MOEA/D-B.

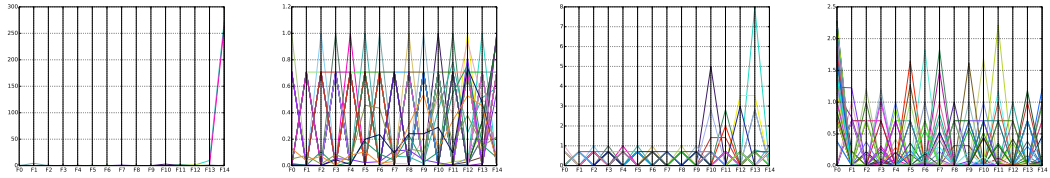


FIGURE A.16: DTLZ(1,2,3,4) benchmarks avec 15 Dimension utilisant U-MOEA/D-B.

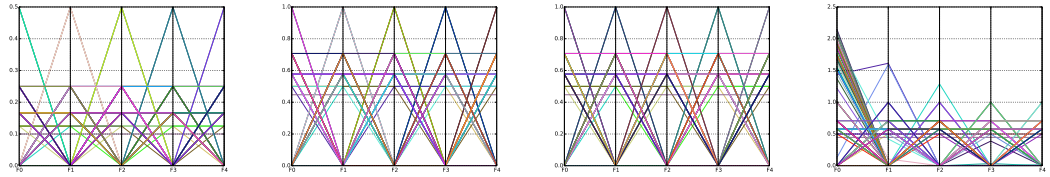


FIGURE A.17: DTLZ(1,2,3,4) benchmark avec 5 Dimension utilisant MOEA/D-T

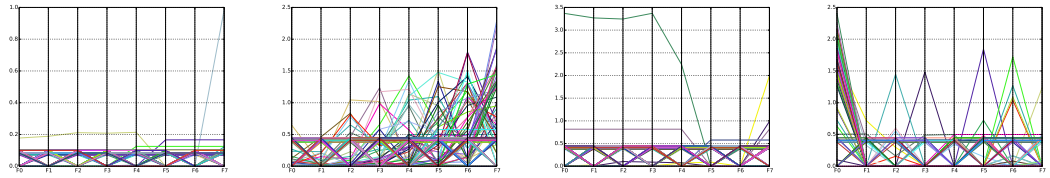


FIGURE A.18: DTLZ(1,2,3,4) benchmark avec 8 Dimension utilisant MOEA/D-T

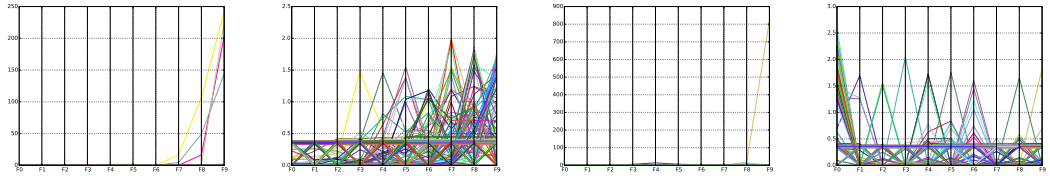


FIGURE A.19: DTLZ(1,2,3,4) benchmark avec 10 Dimension utilisant MOEA/D-T

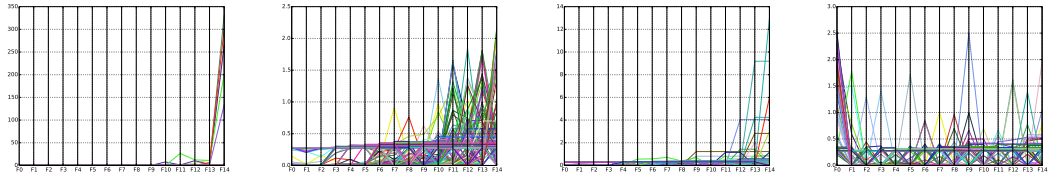


FIGURE A.20: DTLZ(1,2,3,4) benchmark avec 15 Dimension utilisant MOEA/D-T

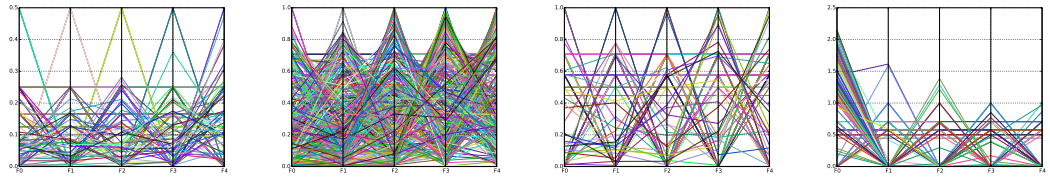


FIGURE A.21: DTLZ(1,2,3,4) benchmark avec 5 Dimension utilisant U-MOEA/D-T

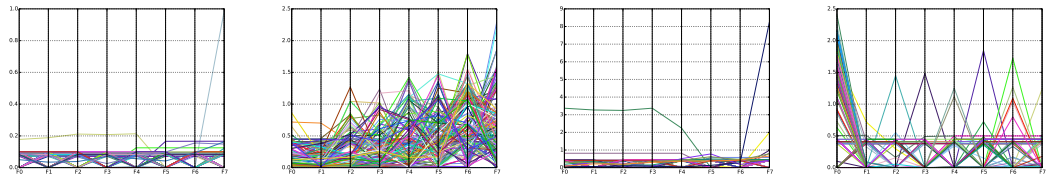


FIGURE A.22: DTLZ(1,2,3,4) benchmark avec 8 Dimension utilisant U-MOEA/D-T

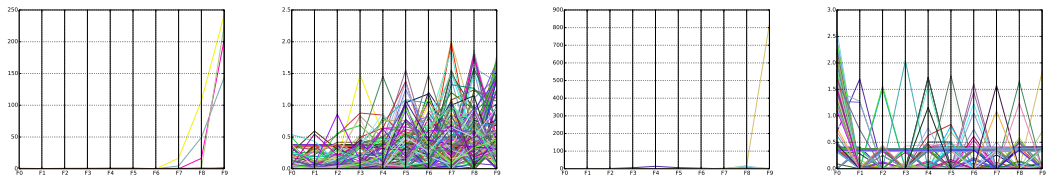


FIGURE A.23: DTLZ(1,2,3,4) benchmark avec 10 Dimension utilisant U-MOEA/D-T

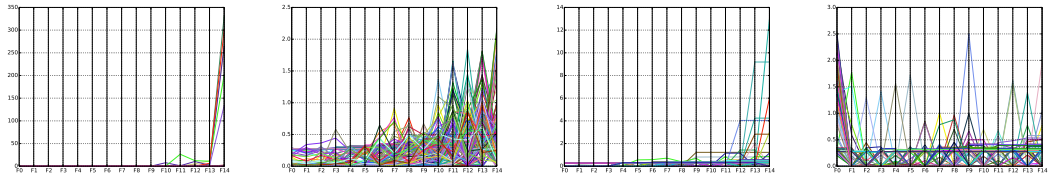


FIGURE A.24: DTLZ(1,2,3,4) benchmark avec 15 Dimension utilisant U-MOEA/D-T

Annexe B

Métriques des résultats pour les problèmes de CEC2009

Dans cette annexe on va présenter les résultats d'exécution de l'algorithme CEDA sur les problèmes de la compétition CEC2009. L'algorithme utilise les deux algorithmes classiques NSGA-II et SPEA2 comme méthode de sélection pour l'algorithme CEDA. Les tableaux résument les moyennes des métriques de qualité utilisé I et IDG sur les différents Benchmark de CEC2009 en appliquant les algorithmes CEDA-NSGA-II et CEDA-SPEA2 avec la comparaison par les algorithmes classiques NSGA-II et SPEA2.

B.1 Les résultats pour les problèmes CEC2009

TABLE B.1: Nombre des nouvelles solutions obtenues pour les CEC2009 Benchmarks

Benchmark	Appel	CEDA-NSGA2	CEDA-SPEA2	NSGA2	SPEA2
CF1	2	$3.41 \cdot 10^{-2}$	$2.83 \cdot 10^{-2}$	$3.24 \cdot 10^{-4}$	$3.06 \cdot 10^{-4}$
CF1	5	$2.05 \cdot 10^{-2}$	$1.78 \cdot 10^{-2}$	$3.25 \cdot 10^{-4}$	$3.03 \cdot 10^{-4}$
CF1	20	$1.98 \cdot 10^{-2}$	$1.65 \cdot 10^{-2}$	$3.25 \cdot 10^{-4}$	$3.07 \cdot 10^{-4}$
CF10	2	$1.94 \cdot 10^{-2}$	$5.55 \cdot 10^{-3}$	$4.92 \cdot 10^{-6}$	$3.31 \cdot 10^{-6}$
CF10	5	$1.42 \cdot 10^{-2}$	$3.32 \cdot 10^{-3}$	$6.83 \cdot 10^{-5}$	$3.64 \cdot 10^{-6}$
CF10	20	$3.27 \cdot 10^{-2}$	$1.19 \cdot 10^{-2}$	$6.05 \cdot 10^{-5}$	$1.63 \cdot 10^{-5}$
CF2	2	$2.56 \cdot 10^{-2}$	$1.27 \cdot 10^{-2}$	$3.29 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
CF2	5	$1.59 \cdot 10^{-2}$	$7.46 \cdot 10^{-3}$	$3.28 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
CF2	20	$1.76 \cdot 10^{-2}$	$1.18 \cdot 10^{-2}$	$3.28 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
CF3	2	$1.29 \cdot 10^{-2}$	$1.87 \cdot 10^{-2}$	$1.99 \cdot 10^{-4}$	$3.27 \cdot 10^{-4}$
CF3	5	$1.25 \cdot 10^{-2}$	$1.79 \cdot 10^{-2}$	$2.77 \cdot 10^{-4}$	$2.78 \cdot 10^{-4}$
CF3	20	$1.38 \cdot 10^{-2}$	$9.65 \cdot 10^{-3}$	$2.67 \cdot 10^{-4}$	$2.7 \cdot 10^{-4}$
CF4	2	$1.9 \cdot 10^{-3}$	$1.77 \cdot 10^{-3}$	$3.28 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
CF4	5	$1.09 \cdot 10^{-3}$	$1.09 \cdot 10^{-3}$	$3.27 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
CF4	20	$1.26 \cdot 10^{-3}$	$1.13 \cdot 10^{-3}$	$3.28 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
CF5	2	0.26	$3.11 \cdot 10^{-2}$	$3.27 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
CF5	5	0.15	$2.09 \cdot 10^{-2}$	$3.27 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
CF5	20	0.16	$3.3 \cdot 10^{-2}$	$3.25 \cdot 10^{-4}$	$3.29 \cdot 10^{-4}$
CF6	2	$2.36 \cdot 10^{-3}$	$1.5 \cdot 10^{-3}$	$3.27 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
CF6	5	$1.48 \cdot 10^{-3}$	$9.96 \cdot 10^{-4}$	$3.27 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
CF6	20	$1.34 \cdot 10^{-3}$	$9.89 \cdot 10^{-4}$	$3.27 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
CF7	2	0.39	0.21	$3.13 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
CF7	5	0.39	0.21	$3.07 \cdot 10^{-4}$	$3.17 \cdot 10^{-4}$
CF7	20	0.37	0.23	$3.2 \cdot 10^{-4}$	$3.13 \cdot 10^{-4}$
CF8	2	0.23	$4.13 \cdot 10^{-2}$	$3.28 \cdot 10^{-4}$	$2.49 \cdot 10^{-4}$
CF8	5	0.19	$4.25 \cdot 10^{-2}$	$3.29 \cdot 10^{-4}$	$2.65 \cdot 10^{-4}$
CF8	20	0.11	$2.78 \cdot 10^{-2}$	$2.59 \cdot 10^{-4}$	$2.48 \cdot 10^{-4}$
CF9	2	0.27	0.18	$3.23 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
CF9	5	0.23	0.15	$3.24 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
CF9	20	0.15	$8.86 \cdot 10^{-2}$	$3.23 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF1	2	$4.01 \cdot 10^{-2}$	$1.09 \cdot 10^{-2}$	$3.26 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$

TABLE B.1: Nombre des nouvelles solutions obtenues pour les CEC2009 Benchmarks

Benchmark	Appel	CEDA-NSGA2	CEDA-SPEA2	NSGA2	SPEA2
UF1	5	$2.44 \cdot 10^{-2}$	$6.26 \cdot 10^{-3}$	$3.27 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF1	20	$2.55 \cdot 10^{-2}$	$7.24 \cdot 10^{-3}$	$3.27 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF10	2	$1.82 \cdot 10^{-2}$	$6.22 \cdot 10^{-2}$	$3.24 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF10	5	$1.68 \cdot 10^{-2}$	$5 \cdot 10^{-2}$	$3.24 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF10	20	$1.37 \cdot 10^{-2}$	$3.04 \cdot 10^{-2}$	$3.24 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF2	2	$7.44 \cdot 10^{-3}$	$4.61 \cdot 10^{-3}$	$3.26 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF2	5	$5.8 \cdot 10^{-3}$	$3.64 \cdot 10^{-3}$	$3.26 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF2	20	$6.04 \cdot 10^{-3}$	$3.43 \cdot 10^{-3}$	$3.26 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF3	2	$7.49 \cdot 10^{-3}$	$8.25 \cdot 10^{-3}$	$5.43 \cdot 10^{-3}$	$4.76 \cdot 10^{-3}$
UF3	5	$6.01 \cdot 10^{-3}$	$6.44 \cdot 10^{-3}$	$5.08 \cdot 10^{-3}$	$4.85 \cdot 10^{-3}$
UF3	20	$2.73 \cdot 10^{-3}$	$3.08 \cdot 10^{-3}$	$4.96 \cdot 10^{-3}$	$4.01 \cdot 10^{-3}$
UF4	2	$8.98 \cdot 10^{-2}$	$2.31 \cdot 10^{-2}$	$3.25 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF4	5	$6.82 \cdot 10^{-2}$	$1.56 \cdot 10^{-2}$	$3.25 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF4	20	$6.34 \cdot 10^{-2}$	$1.61 \cdot 10^{-2}$	$3.25 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF5	2	$9.95 \cdot 10^{-2}$	$1.97 \cdot 10^{-2}$	$1.29 \cdot 10^{-4}$	$1.71 \cdot 10^{-4}$
UF5	5	$5.55 \cdot 10^{-2}$	$1.04 \cdot 10^{-2}$	$1.41 \cdot 10^{-4}$	$1.57 \cdot 10^{-4}$
UF5	20	$2.07 \cdot 10^{-2}$	$1.5 \cdot 10^{-2}$	$1.56 \cdot 10^{-4}$	$1.44 \cdot 10^{-4}$
UF6	2	$1.71 \cdot 10^{-2}$	$1.12 \cdot 10^{-2}$	$3.28 \cdot 10^{-4}$	$3.31 \cdot 10^{-4}$
UF6	5	$1.51 \cdot 10^{-2}$	$1.18 \cdot 10^{-2}$	$3.28 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF6	20	$1.34 \cdot 10^{-2}$	$8.19 \cdot 10^{-3}$	$3.1 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF7	2	$8.07 \cdot 10^{-2}$	$1.05 \cdot 10^{-2}$	$3.27 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF7	5	$6.88 \cdot 10^{-2}$	$9.25 \cdot 10^{-3}$	$3.26 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF7	20	$3.32 \cdot 10^{-2}$	$1.16 \cdot 10^{-2}$	$3.26 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF8	2	0.24	0.17	$3.23 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF8	5	0.22	0.14	$3.23 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF8	20	0.15	$8.88 \cdot 10^{-2}$	$3.23 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF9	2	0.11	$3.5 \cdot 10^{-2}$	$3.24 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF9	5	0.1	$3.42 \cdot 10^{-2}$	$3.24 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
UF9	20	$8.57 \cdot 10^{-2}$	$3.26 \cdot 10^{-2}$	$3.24 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$

TABLE B.2: Moyennes de métrique IGD obtenues pour les CEC2009 Benchmarks

Benchmark	Appel	CEDA-NSGA2	CEDA-SPEA2	NSGA2	SPEA2
CF1	2	$4.63 \cdot 10^{-3}$	$4.54 \cdot 10^{-3}$	$3.85 \cdot 10^{-3}$	$4.33 \cdot 10^{-3}$
CF1	5	$4.14 \cdot 10^{-3}$	$4.16 \cdot 10^{-3}$	$3.9 \cdot 10^{-3}$	$4.29 \cdot 10^{-3}$
CF1	20	$3.93 \cdot 10^{-3}$	$4.15 \cdot 10^{-3}$	$3.94 \cdot 10^{-3}$	$4.28 \cdot 10^{-3}$
CF10	2	0.59	0.76	0.67	0.69
CF10	5	0.59	0.76	0.64	0.63
CF10	20	0.49	0.7	0.65	0.62
CF2	2	$8.02 \cdot 10^{-3}$	$7.46 \cdot 10^{-3}$	$5.38 \cdot 10^{-3}$	$7.51 \cdot 10^{-3}$
CF2	5	$6.79 \cdot 10^{-3}$	$6.87 \cdot 10^{-3}$	$5.33 \cdot 10^{-3}$	$7.09 \cdot 10^{-3}$
CF2	20	$6.02 \cdot 10^{-3}$	$6.49 \cdot 10^{-3}$	$6.24 \cdot 10^{-3}$	$6.66 \cdot 10^{-3}$
CF3	2	$5.87 \cdot 10^{-3}$	$1.32 \cdot 10^{-2}$	0.18	0.18
CF3	5	$5.77 \cdot 10^{-3}$	$1.31 \cdot 10^{-2}$	$7.52 \cdot 10^{-2}$	$7.82 \cdot 10^{-2}$
CF3	20	$5.6 \cdot 10^{-3}$	$1.26 \cdot 10^{-2}$	$2.38 \cdot 10^{-2}$	$2.68 \cdot 10^{-2}$
CF4	2	$7.87 \cdot 10^{-3}$	$1.42 \cdot 10^{-2}$	$6.79 \cdot 10^{-3}$	$1.05 \cdot 10^{-2}$
CF4	5	$6.98 \cdot 10^{-3}$	$1.16 \cdot 10^{-2}$	$6.91 \cdot 10^{-3}$	$1.17 \cdot 10^{-2}$
CF4	20	$6.65 \cdot 10^{-3}$	$9.69 \cdot 10^{-3}$	$6.51 \cdot 10^{-3}$	$1.15 \cdot 10^{-2}$
CF5	2	$1.91 \cdot 10^{-2}$	$2.28 \cdot 10^{-2}$	$1.81 \cdot 10^{-2}$	$2.24 \cdot 10^{-2}$
CF5	5	$1.76 \cdot 10^{-2}$	$2.09 \cdot 10^{-2}$	$1.83 \cdot 10^{-2}$	$2.19 \cdot 10^{-2}$
CF5	20	$1.72 \cdot 10^{-2}$	$1.9 \cdot 10^{-2}$	$1.8 \cdot 10^{-2}$	$2.05 \cdot 10^{-2}$
CF6	2	$1.39 \cdot 10^{-2}$	$1.41 \cdot 10^{-2}$	$1.22 \cdot 10^{-2}$	$1.3 \cdot 10^{-2}$
CF6	5	$1.23 \cdot 10^{-2}$	$1.26 \cdot 10^{-2}$	$1.17 \cdot 10^{-2}$	$1.27 \cdot 10^{-2}$
CF6	20	$1.19 \cdot 10^{-2}$	$1.23 \cdot 10^{-2}$	$1.21 \cdot 10^{-2}$	$1.25 \cdot 10^{-2}$
CF7	2	$1.61 \cdot 10^{-2}$	$1.68 \cdot 10^{-2}$	$1.45 \cdot 10^{-2}$	$1.72 \cdot 10^{-2}$
CF7	5	$1.59 \cdot 10^{-2}$	$1.69 \cdot 10^{-2}$	$1.46 \cdot 10^{-2}$	$1.75 \cdot 10^{-2}$
CF7	20	$1.59 \cdot 10^{-2}$	$1.67 \cdot 10^{-2}$	$1.45 \cdot 10^{-2}$	$1.73 \cdot 10^{-2}$
CF8	2	0.38	0.44	0.37	0.36
CF8	5	0.38	0.44	0.42	0.43
CF8	20	0.38	0.44	0.46	0.44
CF9	2	$3.13 \cdot 10^{-3}$	$2.62 \cdot 10^{-3}$	$2.47 \cdot 10^{-2}$	$2.43 \cdot 10^{-2}$
CF9	5	$3.01 \cdot 10^{-3}$	$2.58 \cdot 10^{-3}$	$1.25 \cdot 10^{-2}$	$1.14 \cdot 10^{-2}$
CF9	20	$2.78 \cdot 10^{-3}$	$2.55 \cdot 10^{-3}$	$5.82 \cdot 10^{-3}$	$5.15 \cdot 10^{-3}$
UF1	2	$3.82 \cdot 10^{-3}$	$4.57 \cdot 10^{-3}$	$3.54 \cdot 10^{-3}$	$4.52 \cdot 10^{-3}$
UF1	5	$3.47 \cdot 10^{-3}$	$4.25 \cdot 10^{-3}$	$3.5 \cdot 10^{-3}$	$4.78 \cdot 10^{-3}$
UF1	20	$3.47 \cdot 10^{-3}$	$4.18 \cdot 10^{-3}$	$3.43 \cdot 10^{-3}$	$4.65 \cdot 10^{-3}$

TABLE B.2: Moyennes de métrique IGD obtenues pour les CEC2009 Benchmarks

Benchmark	Appel	CEDA-NSGA2	CEDA-SPEA2	NSGA2	SPEA2
UF10	2	$5.19 \cdot 10^{-3}$	$5.39 \cdot 10^{-3}$	$8.88 \cdot 10^{-2}$	$8.84 \cdot 10^{-2}$
UF10	5	$5.03 \cdot 10^{-3}$	$5.39 \cdot 10^{-3}$	$3.89 \cdot 10^{-2}$	$3.83 \cdot 10^{-2}$
UF10	20	$4.66 \cdot 10^{-3}$	$5.37 \cdot 10^{-3}$	$1.37 \cdot 10^{-2}$	$1.28 \cdot 10^{-2}$
UF2	2	$1.72 \cdot 10^{-3}$	$2.11 \cdot 10^{-3}$	$1.72 \cdot 10^{-3}$	$1.85 \cdot 10^{-3}$
UF2	5	$1.52 \cdot 10^{-3}$	$1.84 \cdot 10^{-3}$	$1.64 \cdot 10^{-3}$	$2.12 \cdot 10^{-3}$
UF2	20	$1.37 \cdot 10^{-3}$	$1.76 \cdot 10^{-3}$	$1.62 \cdot 10^{-3}$	$1.93 \cdot 10^{-3}$
UF3	2	$1.26 \cdot 10^{-2}$	$1.37 \cdot 10^{-2}$	$2.5 \cdot 10^{-2}$	$2.57 \cdot 10^{-2}$
UF3	5	$1.25 \cdot 10^{-2}$	$1.36 \cdot 10^{-2}$	$1.73 \cdot 10^{-2}$	$1.73 \cdot 10^{-2}$
UF3	20	$1.18 \cdot 10^{-2}$	$1.31 \cdot 10^{-2}$	$1.36 \cdot 10^{-2}$	$1.35 \cdot 10^{-2}$
UF4	2	$1.48 \cdot 10^{-3}$	$1.4 \cdot 10^{-3}$	$1.47 \cdot 10^{-3}$	$1.37 \cdot 10^{-3}$
UF4	5	$1.48 \cdot 10^{-3}$	$1.39 \cdot 10^{-3}$	$1.47 \cdot 10^{-3}$	$1.37 \cdot 10^{-3}$
UF4	20	$1.47 \cdot 10^{-3}$	$1.38 \cdot 10^{-3}$	$1.47 \cdot 10^{-3}$	$1.37 \cdot 10^{-3}$
UF5	2	$6.19 \cdot 10^{-2}$	0.18	$5.34 \cdot 10^{-2}$	0.16
UF5	5	$5.63 \cdot 10^{-2}$	0.16	$5.5 \cdot 10^{-2}$	0.16
UF5	20	$5.54 \cdot 10^{-2}$	0.14	$5.41 \cdot 10^{-2}$	0.15
UF6	2	$9.09 \cdot 10^{-3}$	$8.15 \cdot 10^{-3}$	$9.46 \cdot 10^{-2}$	$9.45 \cdot 10^{-2}$
UF6	5	$9.09 \cdot 10^{-3}$	$8.15 \cdot 10^{-3}$	$4.39 \cdot 10^{-2}$	$4.58 \cdot 10^{-2}$
UF6	20	$9.08 \cdot 10^{-3}$	$7.88 \cdot 10^{-3}$	$1.87 \cdot 10^{-2}$	$2.04 \cdot 10^{-2}$
UF7	2	$1.13 \cdot 10^{-2}$	$1.27 \cdot 10^{-2}$	$2.31 \cdot 10^{-2}$	$2.55 \cdot 10^{-2}$
UF7	5	$1.13 \cdot 10^{-2}$	$1.27 \cdot 10^{-2}$	$1.64 \cdot 10^{-2}$	$1.42 \cdot 10^{-2}$
UF7	20	$1.13 \cdot 10^{-2}$	$1.27 \cdot 10^{-2}$	$1.12 \cdot 10^{-2}$	$1.27 \cdot 10^{-2}$
UF8	2	$3.38 \cdot 10^{-3}$	$2.49 \cdot 10^{-3}$	$1.83 \cdot 10^{-2}$	$1.85 \cdot 10^{-2}$
UF8	5	$3.17 \cdot 10^{-3}$	$2.46 \cdot 10^{-3}$	$9.39 \cdot 10^{-3}$	$9.06 \cdot 10^{-3}$
UF8	20	$2.85 \cdot 10^{-3}$	$2.42 \cdot 10^{-3}$	$4.67 \cdot 10^{-3}$	$4.31 \cdot 10^{-3}$
UF9	2	$3.23 \cdot 10^{-3}$	$2.49 \cdot 10^{-3}$	$1.94 \cdot 10^{-2}$	$1.89 \cdot 10^{-2}$
UF9	5	$3.2 \cdot 10^{-3}$	$2.48 \cdot 10^{-3}$	$9.77 \cdot 10^{-3}$	$8.74 \cdot 10^{-3}$
UF9	20	$3.03 \cdot 10^{-3}$	$2.41 \cdot 10^{-3}$	$5.08 \cdot 10^{-3}$	$3.74 \cdot 10^{-3}$

TABLE B.3: Moyennes de nombre d'évaluations des fonctions obtenues pour les CEC2009 Benchmarks pour chaque appel

Benchmark	Appel	CEDA-SPEA2	CEDA-NSGA2
CF1	1	161.6	92.4
CF1	2	323.73	184.6
CF1	3	486.87	277.4
CF1	4	647.83	370.13
CF1	5	809.33	462.1
CF1	6	971.77	554.53
CF1	7	1,133.53	646.57
CF1	8	1,295.93	738.87
CF1	9	1,457.4	831
CF1	10	1,619.9	923.4
CF1	11	1,781.07	1,016.83
CF1	12	1,942	1,110
CF1	13	2,103.63	1,202.7
CF1	14	2,264.37	1,294.93
CF1	15	2,425.93	1,387.77
CF1	16	2,587.07	1,480.6
CF1	17	2,747.8	1,573.53
CF1	18	2,909	1,666.33
CF1	19	3,069.83	1,758.93
CF1	20	3,231.13	1,851.33
CF2	1	366.37	143.6
CF2	2	719.6	285.1
CF2	3	1,068.57	424.87
CF2	4	1,417.23	561.93
CF2	5	1,786.13	701.87
CF2	6	2,142.37	843.57
CF2	7	2,507.5	980.9
CF2	8	2,859.2	1,121.17
CF2	9	3,216.4	1,260.3
CF2	10	3,575.17	1,397.1
CF2	11	3,932.6	1,536.87
CF2	12	4,293.47	1,674.33

TABLE B.3: Moyennes de nombre d'évaluations des fonctions obtenues pour les CEC2009 Benchmarks pour chaque appel

Benchmark	Appel	CEDA-SPEA2	CEDA-NSGA2
CF2	13	4,654.43	1,814.33
CF2	14	5,010.47	1,950.97
CF2	15	5,370.43	2,093.37
CF2	16	5,731.27	2,232.57
CF2	17	6,082.83	2,372
CF2	18	6,442.87	2,514.8
CF2	19	6,800.53	2,654.4
CF2	20	7,161.97	2,798.93
CF3	1	694.09	295.91
CF3	2	1,337.64	636.18
CF3	3	1,932	927.64
CF3	4	2,499.09	1,240.09
CF3	5	3,168.09	1,553.73
CF3	6	3,789.27	1,844.55
CF3	7	4,381.73	2,159.64
CF3	8	5,026.55	2,500
CF3	9	5,620.27	2,817.73
CF3	10	6,265.55	3,128.09
CF3	11	6,935.82	3,468.91
CF3	12	7,549	3,772.64
CF3	13	8,165.27	4,077.73
CF3	14	8,837.27	4,361.36
CF3	15	9,486.18	4,686.45
CF3	16	10,077.36	5,008
CF3	17	10,750	5,329.55
CF3	18	11,367.36	5,646.82
CF3	19	11,985.55	5,946
CF3	20	12,624.45	6,266.64
CF4	1	754.33	456.4
CF4	2	1,511.1	904.43
CF4	3	2,246.4	1,359.97
CF4	4	2,991.67	1,818.13

TABLE B.3: Moyennes de nombre d'évaluations des fonctions obtenues pour les CEC2009 Benchmarks pour chaque appel

Benchmark	Appel	CEDA-SPEA2	CEDA-NSGA2
CF4	5	3,735.67	2,274.47
CF4	6	4,478.77	2,723.3
CF4	7	5,225.4	3,179.63
CF4	8	5,969.63	3,635.27
CF4	9	6,704.07	4,090.4
CF4	10	7,438.67	4,547.03
CF4	11	8,197.33	5,001.83
CF4	12	8,953.17	5,458.7
CF4	13	9,706.8	5,913.4
CF4	14	10,443.63	6,365.77
CF4	15	11,196.3	6,810.97
CF4	16	11,942.1	7,267.5
CF4	17	12,688.43	7,724.33
CF4	18	13,446.6	8,183.27
CF4	19	14,190.37	8,638.93
CF4	20	14,934.9	9,091.8
CF5	1	393.3	264.27
CF5	2	775.73	522
CF5	3	1,178.2	786.37
CF5	4	1,608.93	1,057.07
CF5	5	2,009.63	1,327.37
CF5	6	2,421.87	1,597.7
CF5	7	2,824.83	1,875.47
CF5	8	3,255.17	2,151.67
CF5	9	3,672.97	2,433.4
CF5	10	4,082.83	2,690.87
CF5	11	4,504.8	2,954.9
CF5	12	4,907.13	3,238.43
CF5	13	5,337.4	3,509.07
CF5	14	5,730.03	3,786.37
CF5	15	6,131.1	4,044.1
CF5	16	6,532.13	4,308.87

TABLE B.3: Moyennes de nombre d'évaluations des fonctions obtenues pour les CEC2009 Benchmarks pour chaque appel

Benchmark	Appel	CEDA-SPEA2	CEDA-NSGA2
CF5	17	6,914.53	4,572.97
CF5	18	7,326.57	4,837.3
CF5	19	7,728.33	5,101.37
CF5	20	8,140.67	5,372.2
CF6	1	742	453.57
CF6	2	1,484.97	904.23
CF6	3	2,227.17	1,356.57
CF6	4	2,969	1,809.63
CF6	5	3,710.77	2,261.3
CF6	6	4,451.87	2,713.3
CF6	7	5,191.53	3,167.17
CF6	8	5,931.23	3,618.1
CF6	9	6,674.17	4,072.87
CF6	10	7,417.63	4,524.07
CF6	11	8,156.67	4,976.17
CF6	12	8,895.63	5,429.2
CF6	13	9,635.8	5,883.6
CF6	14	10,378.43	6,337.9
CF6	15	11,121.07	6,789.3
CF6	16	11,862	7,243.67
CF6	17	12,602.73	7,695.47
CF6	18	13,341.93	8,149.47
CF6	19	14,083.57	8,603.27
CF6	20	14,822.5	9,054.2
CF7	1	257.15	145.75
CF7	2	513.9	281.5
CF7	3	770.85	416.7
CF7	4	1,028.1	553.5
CF7	5	1,285.45	680.8
CF7	6	1,542.6	817.95
CF7	7	1,784.65	953.3
CF7	8	2,026.6	1,080.2

TABLE B.3: Moyennes de nombre d'évaluations des fonctions obtenues pour les CEC2009 Benchmarks pour chaque appel

Benchmark	Appel	CEDA-SPEA2	CEDA-NSGA2
CF7	9	2,283.9	1,217.15
CF7	10	2,541.45	1,344.6
CF7	11	2,798.5	1,481.4
CF7	12	3,056	1,619.2
CF7	13	3,313.3	1,736.8
CF7	14	3,570.6	1,891.3
CF7	15	3,827.55	2,018.05
CF7	16	4,084.1	2,173.6
CF7	17	4,341.15	2,291.35
CF7	18	4,598.05	2,428.3
CF7	19	4,854.8	2,575.2
CF7	20	5,111.5	2,712.15
CF8	1	193.3	97.87
CF8	2	385.9	195.57
CF8	3	578.67	293.53
CF8	4	771.43	391.27
CF8	5	963.9	489.27
CF8	6	1,156.97	587.2
CF8	7	1,349.53	684.9
CF8	8	1,541.7	782.9
CF8	9	1,734.63	880.67
CF8	10	1,926.5	978.37
CF8	11	2,119.2	1,076.03
CF8	12	2,311.97	1,173.97
CF8	13	2,504.13	1,271.93
CF8	14	2,696.03	1,369.83
CF8	15	2,889.03	1,467.6
CF8	16	3,080.9	1,565.23
CF8	17	3,273.37	1,663.6
CF8	18	3,465.87	1,761.73
CF8	19	3,658.83	1,859.5
CF8	20	3,851.37	1,957.23

TABLE B.3: Moyennes de nombre d'évaluations des fonctions obtenues pour les CEC2009 Benchmarks pour chaque appel

Benchmark	Appel	CEDA-SPEA2	CEDA-NSGA2
CF9	1	377.83	168.63
CF9	2	743.67	343.37
CF9	3	1,121.33	514.93
CF9	4	1,485.23	681.67
CF9	5	1,852.63	854.83
CF9	6	2,228.9	1,020.17
CF9	7	2,604.3	1,189.57
CF9	8	2,973.23	1,355
CF9	9	3,348.93	1,521.27
CF9	10	3,742.1	1,684.53
CF9	11	4,126.5	1,844.23
CF9	12	4,505.87	2,015.4
CF9	13	4,879.83	2,188.37
CF9	14	5,252.03	2,347.5
CF9	15	5,629.23	2,515.93
CF9	16	6,010.2	2,685.07
CF9	17	6,385.5	2,839.8
CF9	18	6,778.77	3,010.83
CF9	19	7,163.27	3,171.2
CF9	20	7,541.6	3,349.83
CF10	1	194.43	94.17
CF10	2	389.17	188.2
CF10	3	583.8	282.3
CF10	4	778.37	376
CF10	5	972.6	469.73
CF10	6	1,167.13	563.47
CF10	7	1,362.43	657.03
CF10	8	1,556.83	751.17
CF10	9	1,752.4	844.77
CF10	10	1,947.27	938.6
CF10	11	2,142.43	1,033.43
CF10	12	2,337.47	1,127.77

TABLE B.3: Moyennes de nombre d'évaluations des fonctions obtenues pour les CEC2009 Benchmarks pour chaque appel

Benchmark	Appel	CEDA-SPEA2	CEDA-NSGA2
CF10	13	2,532.53	1,222.67
CF10	14	2,727.3	1,316.77
CF10	15	2,922.27	1,410.5
CF10	16	3,117	1,504.07
CF10	17	3,311.4	1,598.47
CF10	18	3,506.13	1,693
CF10	19	3,701.2	1,787.77
CF10	20	3,896.4	1,882.03
UF1	1	402.37	135.37
UF1	2	803.33	271.67
UF1	3	1,208.53	409.07
UF1	4	1,611.43	545.7
UF1	5	2,014.6	680.53
UF1	6	2,420.7	815.8
UF1	7	2,826.43	953.33
UF1	8	3,234.6	1,089.07
UF1	9	3,641.1	1,225.07
UF1	10	4,040.8	1,360.6
UF1	11	4,438.4	1,496.43
UF1	12	4,847.73	1,632.3
UF1	13	5,259.33	1,768.8
UF1	14	5,663.57	1,903.2
UF1	15	6,077.27	2,037.73
UF1	16	6,485.97	2,173.2
UF1	17	6,890.6	2,307.33
UF1	18	7,295.37	2,445.63
UF1	19	7,698.43	2,580.23
UF1	20	8,107.37	2,713.57
UF2	1	711.33	435.17
UF2	2	1,418.2	868.47
UF2	3	2,133.43	1,301.03
UF2	4	2,849.87	1,736.2

TABLE B.3: Moyennes de nombre d'évaluations des fonctions obtenues pour les CEC2009 Benchmarks pour chaque appel

Benchmark	Appel	CEDA-SPEA2	CEDA-NSGA2
UF2	5	3,556.03	2,171.07
UF2	6	4,265.97	2,600.27
UF2	7	4,981.2	3,035.33
UF2	8	5,691.73	3,469.7
UF2	9	6,404.9	3,903
UF2	10	7,117.53	4,334.8
UF2	11	7,829.4	4,767.9
UF2	12	8,543.9	5,200.3
UF2	13	9,256.53	5,634.77
UF2	14	9,971.43	6,070.47
UF2	15	10,682.8	6,506.6
UF2	16	11,401	6,941.53
UF2	17	12,110.6	7,373.9
UF2	18	12,825.83	7,807.7
UF2	19	13,535.13	8,239.7
UF2	20	14,247.23	8,675.83
UF3	1	790.67	491.07
UF3	2	1,580.33	984.27
UF3	3	2,369.47	1,477
UF3	4	3,158.37	1,969.23
UF3	5	3,948.73	2,461.67
UF3	6	4,738.9	2,954.53
UF3	7	5,528.17	3,447.57
UF3	8	6,317.37	3,937.73
UF3	9	7,107.1	4,430.7
UF3	10	7,897.13	4,922.8
UF3	11	8,686.33	5,415.03
UF3	12	9,476.1	5,905.2
UF3	13	10,265.8	6,396.9
UF3	14	11,056.17	6,889.2
UF3	15	11,847.47	7,380.3
UF3	16	12,636.47	7,871.83

TABLE B.3: Moyennes de nombre d'évaluations des fonctions obtenues pour les CEC2009 Benchmarks pour chaque appel

Benchmark	Appel	CEDA-SPEA2	CEDA-NSGA2
UF3	17	13,425.53	8,364.07
UF3	18	14,214.83	8,856.63
UF3	19	15,004.37	9,349.97
UF3	20	15,794.53	9,842.97
UF4	1	200	99
UF4	2	400	197.93
UF4	3	600	296.9
UF4	4	800	395.93
UF4	5	999.97	495.07
UF4	6	1,199.97	594.17
UF4	7	1,399.97	693.07
UF4	8	1,599.93	792.1
UF4	9	1,799.93	890.93
UF4	10	1,999.93	989.97
UF4	11	2,199.93	1,088.93
UF4	12	2,399.93	1,187.93
UF4	13	2,599.93	1,286.87
UF4	14	2,799.93	1,385.7
UF4	15	2,999.93	1,484.7
UF4	16	3,199.9	1,583.77
UF4	17	3,399.83	1,682.87
UF4	18	3,599.8	1,781.67
UF4	19	3,799.8	1,880.6
UF4	20	3,999.77	1,979.47
UF5	1	520.6	276.63
UF5	2	1,041.6	557.07
UF5	3	1,562.93	840.63
UF5	4	2,088.37	1,106.53
UF5	5	2,585.93	1,380.13
UF5	6	3,108.87	1,659.97
UF5	7	3,618	1,940.13
UF5	8	4,137.33	2,220.3

TABLE B.3: Moyennes de nombre d'évaluations des fonctions obtenues pour les CEC2009 Benchmarks pour chaque appel

Benchmark	Appel	CEDA-SPEA2	CEDA-NSGA2
UF5	9	4,650.53	2,491.17
UF5	10	5,171.53	2,767.57
UF5	11	5,681.53	3,048.2
UF5	12	6,193.43	3,318.7
UF5	13	6,722.93	3,596.93
UF5	14	7,246.23	3,875.27
UF5	15	7,771.4	4,132.53
UF5	16	8,296.4	4,403.93
UF5	17	8,802.5	4,676.07
UF5	18	9,306.37	4,945.5
UF5	19	9,828.53	5,231.97
UF5	20	10,363.17	5,507.7
UF6	1	419.56	426.89
UF6	2	837.78	865.44
UF6	3	1,258.22	1,326.33
UF6	4	1,708.11	1,771.78
UF6	5	2,127.22	2,215.44
UF6	6	2,538.56	2,639.67
UF6	7	2,956.11	3,083.33
UF6	8	3,369.22	3,527.89
UF6	9	3,751.33	3,935.22
UF6	10	4,199.89	4,361.78
UF6	11	4,579.22	4,797.22
UF6	12	5,030.22	5,237.56
UF6	13	5,446.56	5,684.78
UF6	14	5,861.11	6,128.67
UF6	15	6,275.67	6,570.11
UF6	16	6,689.89	7,018.44
UF6	17	7,102.22	7,462.22
UF6	18	7,549.67	7,903.11
UF6	19	7,998.33	8,361.78
UF6	20	8,413.67	8,822.78

TABLE B.3: Moyennes de nombre d'évaluations des fonctions obtenues pour les CEC2009 Benchmarks pour chaque appel

Benchmark	Appel	CEDA-SPEA2	CEDA-NSGA2
UF7	1	383.63	143.2
UF7	2	770.33	287.77
UF7	3	1,152.87	433.7
UF7	4	1,533.17	580
UF7	5	1,918	725.77
UF7	6	2,308.53	869.23
UF7	7	2,702.97	1,014.93
UF7	8	3,085.3	1,158.47
UF7	9	3,465	1,304
UF7	10	3,848.37	1,452.07
UF7	11	4,232.17	1,598.53
UF7	12	4,613.6	1,744.17
UF7	13	4,995.57	1,889.87
UF7	14	5,378.67	2,036.13
UF7	15	5,764.9	2,179.77
UF7	16	6,147.27	2,324.3
UF7	17	6,536.77	2,468.8
UF7	18	6,919.23	2,613.43
UF7	19	7,305.83	2,755.93
UF7	20	7,692.97	2,902.43
UF8	1	377.17	159.67
UF8	2	756.7	320
UF8	3	1,131.9	480.53
UF8	4	1,502.23	642.4
UF8	5	1,876.13	801
UF8	6	2,246.87	959.03
UF8	7	2,624.57	1,122.73
UF8	8	2,993.47	1,282.53
UF8	9	3,368.3	1,445
UF8	10	3,739.17	1,602.4
UF8	11	4,114.23	1,763.37
UF8	12	4,492.83	1,923.63

TABLE B.3: Moyennes de nombre d'évaluations des fonctions obtenues pour les CEC2009 Benchmarks pour chaque appel

Benchmark	Appel	CEDA-SPEA2	CEDA-NSGA2
UF8	13	4,867.23	2,085.07
UF8	14	5,237.43	2,248.57
UF8	15	5,615.83	2,405.87
UF8	16	5,991.07	2,567.13
UF8	17	6,360.9	2,726.6
UF8	18	6,735.17	2,887.93
UF8	19	7,106.4	3,050.13
UF8	20	7,477.43	3,212.4
UF9	1	501.1	123.67
UF9	2	1,004.2	247.53
UF9	3	1,505.07	372.13
UF9	4	2,011.37	497.6
UF9	5	2,513.83	620.3
UF9	6	3,013.7	743.57
UF9	7	3,513.83	868.4
UF9	8	4,014.3	994.53
UF9	9	4,519.13	1,118.27
UF9	10	5,028.47	1,242.67
UF9	11	5,536.8	1,366.73
UF9	12	6,047.6	1,490.37
UF9	13	6,553.17	1,613.97
UF9	14	7,057.03	1,737.33
UF9	15	7,561.07	1,857.57
UF9	16	8,062.4	1,980.2
UF9	17	8,567.63	2,103.97
UF9	18	9,069.23	2,229.33
UF9	19	9,569.63	2,353.93
UF9	20	10,075.1	2,479.57
UF10	1	588.07	306.2
UF10	2	1,175.4	613.63
UF10	3	1,759.03	922.33
UF10	4	2,353.93	1,230.1

TABLE B.3: Moyennes de nombre d'évaluations des fonctions obtenues pour les CEC2009 Benchmarks pour chaque appel

Benchmark	Appel	CEDA-SPEA2	CEDA-NSGA2
UF10	5	2,941.3	1,537.83
UF10	6	3,528.33	1,842.33
UF10	7	4,114.63	2,154.2
UF10	8	4,702.9	2,463.87
UF10	9	5,288.77	2,774.07
UF10	10	5,879.03	3,079.4
UF10	11	6,467.67	3,392
UF10	12	7,051.8	3,695.63
UF10	13	7,634.73	4,010.1
UF10	14	8,220.2	4,321.4
UF10	15	8,803.33	4,626.7
UF10	16	9,390.93	4,939.23
UF10	17	9,980.8	5,247.33
UF10	18	10,566.93	5,553.9
UF10	19	11,152.03	5,862.03
UF10	20	11,737.03	6,166.47