TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ALGORITHMES

INTRODUCTION

Depuis de nombreuses années, les chercheurs s'intéressent à des problèmes complexes qui se composent de plusieurs objectifs à atteindre simultanément. La résolution de tels problèmes représente encore aujourd'hui un défi important puisqu'ils se retrouvent dans de nombreux domaines tels que la médecine, la chimie ou encore l'industrie. Les objectifs associés à ces problèmes sont souvent contradictoires, de sorte qu'il est très difficile de trouver une unique solution optimisant chacun des objectifs. Dans un tel contexte, il convient généralement de chercher les solutions qui représentent les meilleurs compromis entre les objectifs. En recherche opérationnelle, ce genre de problème est appelé problème multi-objectifs et l'ensemble de solutions de meilleurs compromis est communément nommé ensemble de solutions *Pareto optimales* en référence à son créateur Vilfredo Pareto (Pareto, 1876).

Une majorité de problèmes, en plus d'être multi-objectifs, sont également des problèmes NP-Difficiles et ne sont pas résolubles de manière exacte en raison d'un temps de calcul excessif. Néanmoins, des méthodes approchées telles que les métaheuristiques permettent de proposer des solutions acceptables à ces problèmes en un temps qualifié de raisonnable. Les métaheuristiques s'inspirent de phénomènes naturels réels et se classent en plusieurs familles. Parmi les métaheuristiques les plus étudiées dans la littérature se trouvent l'optimisation par colonies de fourmis (Dorigo et Gambardella, 1997), les algorithmes évolutionnaires (Goldberg, 1989) basés sur la théorie de l'évolution proposée par Darwin ou encore les algorithmes par essaim particulaire (Eberhart et Kennedy, 1995). Ces métaheuristiques à base de population sont particulièrement adaptées pour résoudre des problèmes multi-objectifs puisqu'elles permettent d'approximer l'ensemble des solutions Pareto optimales en une seule exécution.

Pour résoudre des problèmes réels à grande échelle, le temps de résolution des métaheuristiques multi-objectifs peut demeurer prohibitif, notamment car ces méthodes ne cherchent pas à approximer une unique solution optimale mais l'ensemble des meilleurs compromis. De ce fait, des alternatives doivent être employées pour réduire le temps d'exécution et ainsi permettre la résolution efficace de problèmes multi-objectifs de grande taille. L'émergence des architectures parallèles qui sont de plus en plus puissantes et accessibles de nos jours a permis, en ce sens, de rendre la parallélisation des métaheuristiques populaire. En plus de réduire le temps pour optimiser un problème, les modèles parallèles pour les métaheuristiques multi-objectifs ont pour but (Talbi *et al.*, 2008) (i) d'améliorer la qualité des solutions obtenues en fournissant une approximation plus proche des meilleures solutions, mais également en proposant un ensemble de solutions mieux diversifié (ii) de répondre à des instances de problème plus grandes qui ne peuvent pas être traitées sur des processeurs monocœur (iii) d'améliorer la robustesse des approches sur différents types de problème. En suivant ces objectifs et depuis les années 2000, plusieurs méthodes de parallélisation ont été proposées pour améliorer les performances des métaheuristiques multi-objectifs. Bien que ces approches aient apporté une contribution significative au domaine de l'optimisation multi-objectifs (Talbi, 2018), elles restent perfectibles en ce qui concerne leur extensibilité et l'exploitation efficace de multiples unités de calcul.

L'objectif principal de la thèse est de contribuer à amener les modèles parallèles conçus pour les métaheuristiques multi-objectifs à exploiter efficacement les architectures de calcul haute performance. Des stratégies et mécanismes sont ainsi proposés au sein d'un nouveau modèle parallèle basé sur le paradigme en îles. Un algorithme de partitionnement original est notamment utilisé pour répartir la recherche des solutions Pareto optimales sur les différentes îles. Dans le but d'évaluer les performances du modèle, une étude expérimentale minutieuse est effectuée et permet de valider chaque composante dans la résolution de deux problèmes combinatoires classiques. La qualité des solutions obtenues par la nouvelle approche est analysée par le biais de différentes métriques et les résultats sont comparés aux modèles de la littérature. Pour atteindre l'objectif principal, cette thèse est organisée en cinq chapitres.

Le premier chapitre est consacré à la résolution des problèmes multi-objectifs par les métaheuristiques. Tout d'abord, les enjeux et particularités de ces problèmes sont exposés ce qui permet de situer les difficultés qu'amène l'optimisation de plusieurs objectifs. Un état de l'art structuré des métaheuristiques multi-objectifs est ensuite proposé. Les métriques qui permettent d'évaluer les performances d'un algorithme de résolution multi-objectifs sont

également décrites puisqu'elles sont utilisées durant les diverses expérimentations de cette recherche.

Dans le deuxième chapitre, une introduction au parallélisme est tout d'abord effectuée en présentant notamment les principales architectures matérielles et les environnements logiciels correspondants. La combinaison des deux domaines étudiés dans cette recherche, c'est-à-dire l'optimisation multi-objectifs et la programmation parallèle, est décrite à travers une revue de littérature des modèles parallèles consacrés aux métaheuristiques multi-objectifs. Elle se focalise sur les deux grands paradigmes : le paradigme maître-esclave et le paradigme en îles. Les avantages et les limites des différents modèles parallèles sont ainsi identifiés. Ce chapitre permet de contextualiser la thèse et de présenter les objectifs auxquels elle doit répondre.

Le troisième chapitre détaille le nouveau modèle parallèle pour les métaheuristiques multi-objectifs en présentant indépendamment chaque composante de celui-ci. Une stratégie *diviser pour régner* permet de répartir efficacement les processeurs disponibles dans l'espace objectif. Le modèle a pour but d'optimiser les performances au niveau du parallélisme, mais il doit également fournir des solutions de meilleure qualité que les modèles de la littérature.

Dans le quatrième chapitre, une comparaison empirique de quelques modèles parallèles pour les algorithmes évolutionnaires multi-objectifs est effectuée. Afin de contextualiser les expérimentations, les problèmes combinatoires et l'algorithme évolutionnaire multi-objectifs GISMOO (Zinflou *et al.*, 2012) sur lesquels sont basées toutes les expérimentations sont décrits. La deuxième partie de ce chapitre présente une première comparaison empirique d'une version particulière du modèle proposé et des principaux modèles en îles existants, adaptés à l'algorithme GISMOO. Elle vise à comparer les modèles parallèles existants et à analyser certaines composantes spécifiques du modèle. Celle-ci constitue les prémisses de travaux concernant une expérimentation complète du nouveau modèle proposé.

Le dernier chapitre met en évidence les performances du modèle comparativement aux résultats de la littérature dans la résolution de deux problèmes combinatoires multi-objectifs classiques. Pour ce faire, le calcul des métriques pendant la phase d'expérimentation permet d'analyser sa capacité de convergence vers les solutions Pareto optimales, son habileté à fournir un ensemble de solutions diversifiées et son extensibilité sur de multiples cœurs de processeur.

Finalement, la dernière section de ce document conclut cette recherche par une analyse

générale de l'ensemble des travaux réalisés. En résumé, cette thèse montre que le modèle parallèle, nommé APM-MOEA, permet d'améliorer et d'accélérer un algorithme évolutionnaire multi-objectifs. Sur les bases d'une parallélisation de GISMOO et dans la résolution de deux problèmes combinatoires, APM-MOEA fournit des solutions de meilleure qualité que des modèles parallèles proposés dans la littérature. Le nouveau modèle se révèle également compétitif vis-à-vis d'algorithmes de résolution adaptés spécifiquement aux problèmes traités. Sa capacité de convergence vers le front Pareto et son aptitude à fournir des ensembles de solutions diversifiés est montrée à travers différentes expérimentations. De plus, l'augmentation du nombre de coeurs de processeur permet à APM-MOEA de proposer de meilleures solutions en explorant plus efficacement l'espace objectif. Enfin, les communications asynchrones diminuent les coûts du parallélisme ce qui permet de réduire le temps d'exécution du modèle.

CHAPITRE 1

L'OPTIMISATION MULTI-OBJECTIFS PAR LES MÉTAHEURISTIQUES

Beaucoup de problèmes pratiques nécessitent l'optimisation de plusieurs objectifs qui sont conflictuels, car essayer d'en améliorer un risque d'en dégrader un autre. En recherche opérationnelle, ces problèmes sont appelés problèmes multi-objectifs. Depuis les années 70, les chercheurs s'intéressent à ces problèmes que ce soit dans des domaines comme l'industrie (Stewart *et al.*, 2008), la médecine (Ehrgott et Winz, 2008) ou encore l'économie (Metaxioti et Liagkouras, 2012). De ce fait, le développement d'algorithmes multi-objectifs pour les résoudre devient une nécessité. La principale utilisation de l'optimisation multi-objectifs dans la littérature est naturellement de résoudre de tels problèmes. Cependant, comme le soulignent Handl et Knowles (2008), il existe également une autre utilisation de l'optimisation multi-objectifs qui consiste à transformer la nature d'un problème uni-objectif pour le rendre multi-objectifs et faciliter sa résolution. Par exemple, il est possible de retirer les contraintes d'un problème en associant chacune d'entre elles à une nouvelle fonction objectif (Mezura-Montes et Coello, 2008a).

Afin de mieux comprendre les enjeux et les particularités de l'optimisation multi-objectifs, les concepts importants de ce domaine sont décrits dans la première partie de ce chapitre. La deuxième partie, quant à elle, expose différentes métriques qui permettent de mesurer la qualité d'un ensemble de solutions obtenues. Ensuite, une classification concepteur des méthodes de résolution est introduite et chaque catégorie de méthodes est brièvement présentée. Les deux dernières parties se focalisent principalement sur les métaheuristiques basées sur une population qui utilisent directement les notions de Pareto. Les techniques et stratégies utilisées par chacun des algorithmes sont décrites.

1.1 DÉFINITIONS ET CONCEPTS IMPORTANTS DE L'OPTIMISATION MULTI-OBJECTIFS

1.1.1 FORMULATION D'UN PROBLÈME MULTI-OBJECTIFS

Un problème d'optimisation uni-objectif cherche à minimiser une fonction objectif f dans un espace de recherche S et sous certaines contraintes. Cette fonction associe à chaque solution réalisable, c'est-à-dire qui respecte les contraintes, une valeur de fonction objectif qui est généralement un réel. Un problème peut être associé à des variables de décision qui forment l'espace de recherche. Si le nombre de variables est au moins de deux, on peut alors parler de vecteur de décision. Ces variables de décision sont divisées en deux types : les variables discrètes et les variables continues. Les variables discrètes peuvent prendre des valeurs appartenant à un ensemble fini ou à un ensemble infini dénombrable (e.g. : ensemble des entiers naturels). Au contraire, les variables continues peuvent prendre une infinité de valeurs comprises dans un intervalle défini (e.g. ensemble des réels compris dans $[0,5]$). En fonction du type de variables utilisées, on parlera de problème à variables continues (problème continu), de problème à variables discrètes (problème combinatoire) ou encore de problème mixte qui utilise les deux types de variables.

Il faut noter que tout problème de maximisation peut se ramener à un problème de minimisation puisque $max(f(x)) = -min(-(f(x)))$. En outre, résoudre un problème de minimisation uni-objectif revient à chercher l'optimum global de la fonction objectif, c'est-à-dire trouver une solution s^* qui a la plus petite valeur de fonction objectif. Autrement dit, trouver s^*, tel que $\forall s \in S, f(s^*) \leq f(s)$. Cette recherche est possible, car il existe une relation d'ordre clairement définie dans l'ensemble $\mathbb{R}$. Dans un problème multi-objectifs, contrairement à un problème uni-objectif, plusieurs fonctions objectif sont à minimiser. La formulation mathématique est présentée à la Définition 1.

Définition 1 *(Problème multi-objectifs)*

$$\begin{aligned} &min \quad F(x) = ((f_1(x), f_2(x), ..., f_M(x))) \\ &s.c \quad x \in X \end{aligned} \tag{1.1}$$

avec $M \geq 2$ le nombre de fonctions objectif à minimiser et X l'ensemble des solutions réalisables sous les contraintes du problème.

À chaque vecteur de décision est associé un vecteur objectif $y = (y_1, y_2, ..., y_M)$ où $y_i = f_i(x)$. Le but de la résolution d'un problème multi-objectifs est de minimiser ce vecteur objectif. L'utopie serait de trouver le vecteur qui optimise chacune des fonctions objectif. Or, dans la plupart des problèmes de la vie réelle, ce vecteur n'existe pas, car les objectifs sont conflictuels. Par la suite, l'espace contenant les vecteurs objectif sera appelé espace objectif. De plus, alors que dans un problème uni-objectif, une relation d'ordre est clairement définie pour les valeurs de la fonction objectif (ensemble $\mathbb{R}$), cette relation n'existe pas pour des vecteurs objectif. Il est donc nécessaire d'introduire différents concepts qui permettent tout de même de comparer des vecteurs objectif. Ces concepts sont utilisés par certains algorithmes de résolution présentés par la suite.

Un exemple concret de problème à plusieurs objectifs conflictuels peut être donné en considérant l'achat d'une voiture puisque le client voudra à la fois minimiser certains objectifs comme le prix, la consommation d'essence moyenne et maximiser d'autres aspects tels que le confort ou la puissance moteur. En prenant en compte seulement les objectifs de prix et de puissance, un problème bi-objectif est obtenu et des exemples de solutions sont proposés à la Figure 1.1. Si le client est uniquement intéressé par l'aspect financier, alors il choisira la voiture la moins chère, c'est-à-dire la petite voiture citadine. Au contraire, si la puissance moteur est son principal intérêt, alors il s'orientera vers la voiture de course. Ces deux solutions représentent les solutions extrêmes du problème puisqu'ils optimisent indépendamment un des deux objectifs. Les autres solutions existantes sont des solutions de compromis et permettent d'optimiser en même temps les deux objectifs avec des pondérations différentes en fonction des préférences du client.

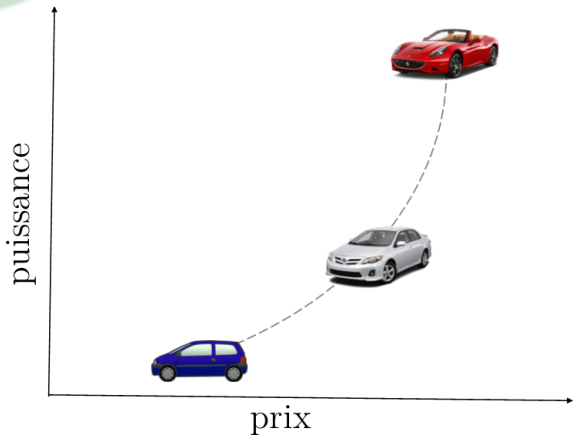


Figure 1.1: Exemple d'un problème d'achat de voiture

1.1.2 DOMINANCE PARETO

Dans la résolution d'un problème multi-objectifs, la notion d'optimalité de l'optimisation uni-objectif est modifiée. Pour expliquer ce nouveau concept, il convient tout d'abord d'introduire la relation de dominance qui a été proposée par l'économiste italien Pareto (1876) et qui permet de comparer deux vecteurs objectif pour un problème de minimisation.

Définition 2 *(Dominance au sens Pareto) On dit qu'un vecteur objectif* $\vec{u} = \{u_1, u_2, ..., u_M\}$ *domine, au sens Pareto, un vecteur* $\vec{v} = \{v_1, v_2, ..., v_M\}$ *si et seulement si* $\forall i \in \{1....M\}, u_i \leq v_i$ *et* $\exists j \in \{1....M\}$ *tel que* $u_j < v_j$*. Cela se traduit par* $\vec{u} \prec \vec{v}$ *.*

Cette relation de dominance n'est pas une relation d'ordre total. En effet, deux vecteurs objectif n'ont pas forcément une relation de dominance établie entre eux. On parle alors d'indifférence entre les deux solutions. Elles ne pourront être différenciées que par les préférences du décideur. À partir de la relation de dominance, il est alors possible de définir ce qu'est un ensemble de solutions Pareto optimales.

Définition 3 *(Ensemble de solutions Pareto optimales) Une solution associée au vecteur objectif* $\vec{u} = \{u_1, u_2, ..., u_M\}$ *est dite non-dominée ou Pareto optimale s'il n'existe aucune autre solution dans l'espace de recherche qui la domine. La totalité des solutions non-dominées forme l'ensemble Pareto optimal, appelé également front Pareto.*

1.1.3 POINT IDÉAL ET POINT NADIR

Le vecteur objectif Idéal (ou point Idéal) z^* est un vecteur utopique qui minimise chacune des fonctions objectif. $z^* = \{f_1^*, f_2^*, ...f_M^*\}$ où f_i^* est le minimum de la fonction objectif f_i. Bien que ce vecteur soit utopique, il peut être utilisé comme référence dans certaines approches de résolution.

Au contraire, le vecteur Nadir z^n (ou point Nadir) est le vecteur objectif qui contient les limites supérieures de chaque objectif pour les solutions du front Pareto ou pour un ensemble de solutions non-dominées. Des exemples de point Nadir et de point Idéal sont présentés à la Figure 1.2. Plusieurs algorithmes de résolution ou métriques utilisent le vecteur Idéal et le vecteur Nadir afin de normaliser chaque objectif. La normalisation est alors donnée par la

formule de l'Équation 1.2.

$$f_i^{\text{norm}} = \frac{f_i - z_i^*}{z_i^n - z_i^*} \tag{1.2}$$

La Figure 1.2 représente un exemple d'espace objectif pour un problème de minimisation à deux objectifs. Les solutions représentées par un disque noir forment le front Pareto et sont des solutions non-dominées alors que les solutions représentées par un losange sont des solutions dominées. Le point Idéal et le point Nadir sont également exposés et sont symbolisés par des disques gris.

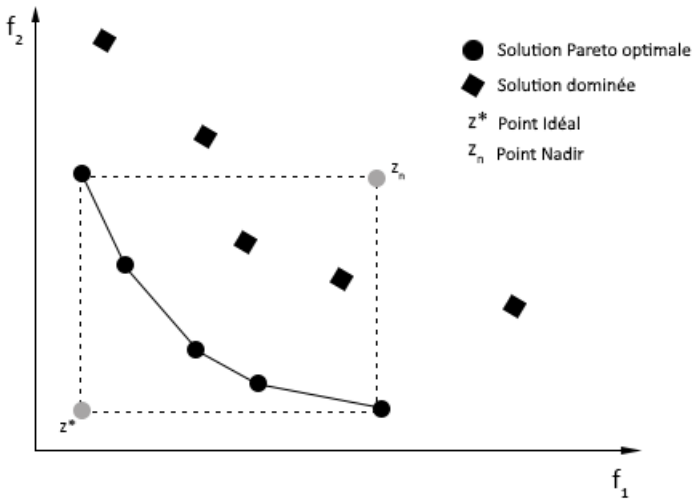
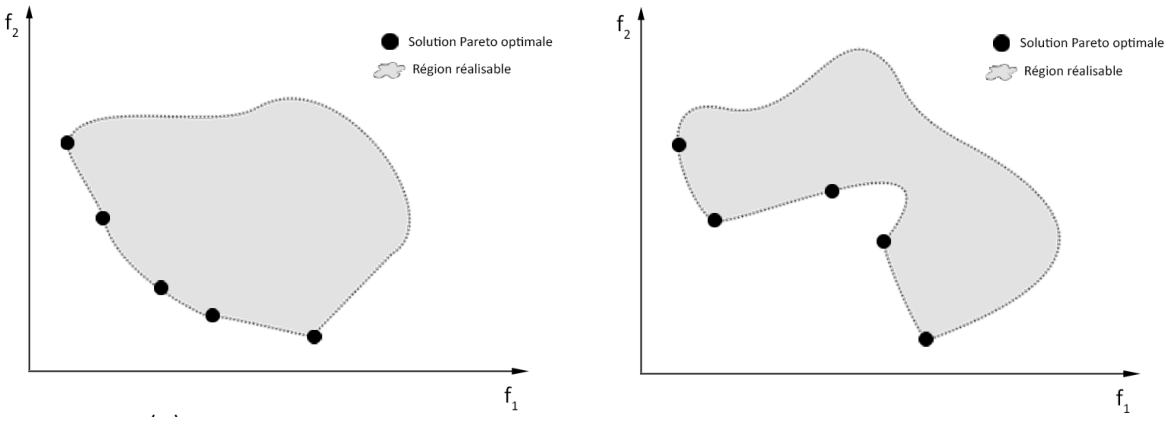


Figure 1.2: Exemple d'un espace objectif pour un problème de minimisation bi-objectif avec le point Idéal et le point Nadir

1.1.4 CARACTÉRISTIQUES DU FRONT PARETO

La facilité d'un algorithme à obtenir des solutions bien réparties dans l'espace objectif dépend, en partie, des caractéristiques du front Pareto. La structure de ce dernier peut être convexe ou non-convexe (concave) et continue ou discontinue. Pour un problème à deux objectifs, un front Pareto est dit convexe si, dans l'espace objectif, tout segment qui lie deux solutions Pareto optimales est inclus dans la région des solutions réalisables. Cette définition peut être étendue à plus de deux objectifs en augmentant le nombre de dimensions. Les Figures 1.3a et 1.3b présentent respectivement un exemple de front convexe et un exemple de front

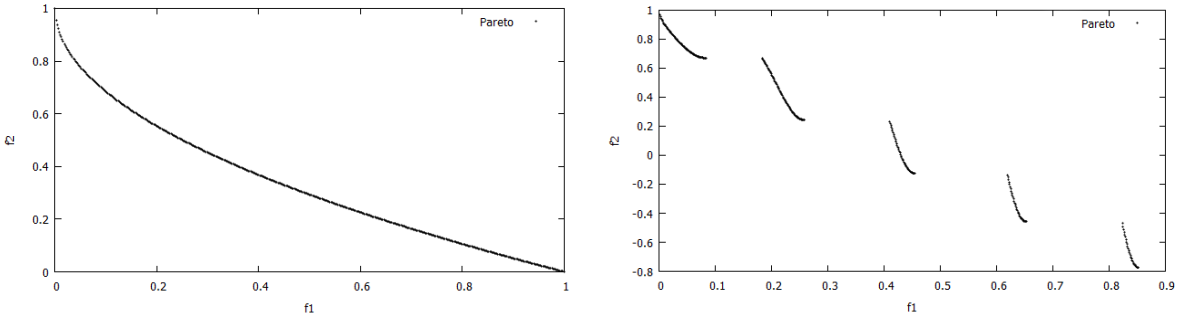
concave pour un problème de minimisation bi-objectif.



(a) Front Pareto convexe **(b) Front Pareto concave**

Figure 1.3: Exemples de front convexe et concave d'un problème bi-objectif

La structure d'un front Pareto peut être également continue ou discontinue. Les algorithmes qui ne prennent pas assez en compte la diversité dans leur résolution auront plus de difficulté à trouver un ensemble de solutions Pareto optimales acceptable quand le front Pareto est discontinu. De même, un algorithme qui utilise les notions de voisinage (recherche avec tabous, recuit simulé...) est plus à risque d'être en difficulté face à des fronts discontinus (Talbi, 2009). Les Figures 1.4a et 1.4b présentent respectivement les fronts Pareto pour les problèmes continus ZDT1 et ZTD3 (Zitzler *et al.*, 2000). Le problème ZDT1 engendre un front Pareto continu alors que le problème ZDT3 engendre un front Pareto discontinu.



(a) ZDT1 **(b) ZDT3**

Figure 1.4: Front Pareto des problèmes ZDT1 et ZDT3

1.2 MÉTRIQUES POUR MESURER LA QUALITÉ D'UN ENSEMBLE DE SOLUTIONS

Les performances d'un algorithme de résolution uni-objectif s'évaluent essentiellement par sa capacité à fournir une solution qui minimise la fonction objectif. Il est plus difficile d'évaluer l'efficacité d'une heuristique multi-objectifs puisqu'il est nécessaire d'analyser un ensemble de solutions associées à des vecteurs objectif. Le but des approches de résolution est de trouver une approximation la plus proche possible du front Pareto et qui couvre une grande partie de l'espace objectif. Comme il n'est pas aisé d'évaluer ces deux aspects en même temps à travers une unique métrique, le calcul de plusieurs métriques est généralement nécessaire. Ainsi, dans la littérature, des métriques unaires et binaires ont été proposées pour évaluer les performances de méthodes de résolution et se focalisent soit sur la convergence soit sur la diversité des solutions. Il est important de noter que le calcul des métriques s'effectue dans la plupart des cas une fois l'exécution de l'algorithme de résolution terminé et donc n'altère pas le processus de recherche ni le temps de calcul. Par la suite, les métriques qui seront utilisées dans le cadre de cette thèse sont présentées. Elles constituent une liste non exhaustive des méthodes d'évaluation existantes et figurent parmi les plus utilisées dans la littérature.

1.2.1 MÉTRIQUES GD ET IGD

La métrique de convergence *GD* (*Generational Distance*), proposée par Van Veldhuizen et Lamont (2000), permet de mesurer la convergence d'un ensemble A de solutions obtenues vers un ensemble de référence R, qui est dans l'idéal le front Pareto. Pour chaque solution de A, la distance Euclidienne minimale par rapport à une des solutions de R est calculée. La mesure de convergence $GD(A,R)$ correspond alors à la moyenne de ces distances. Elle est formulée par l'Équation 1.3.

$$GD(A,R) = \frac{1}{|A|} \sum_{a \in A} \left(\min \{ \text{dist}(a,r) | r \in R \} \right) \tag{1.3}$$

avec $|A|$ le nombre de solutions contenues dans A et $\text{dist}(a,r)$ la distance euclidienne entre les solutions a et r.

La métrique *IGD* (*Inverted Generational Distance*) (Coello et Reyes Sierra, 2004) reprend la métrique *GD* en inversant l'ensemble à évaluer A avec l'ensemble de référence R. Cette me-

sure est moins sensible aux nombres de solutions trouvées dans l'ensemble A. La formulation de $IGD(A,R)$ est donnée à l'Équation 1.4.

$$IGD(A,R) = \frac{1}{|R|} \sum_{r \in R} (\min \{\text{dist}(r,a) | a \in A\}) \tag{1.4}$$

avec $|R|$ le nombre de solutions contenues dans R et $\text{dist}(r,a)$ la distance euclidienne entre les solutions r et a.

1.2.2 L'HYPER-VOLUME $\mathscr{S}$

Zitzler et Thiele (1998) ont proposé la métrique nommée hyper-volume et qui est notée $\mathscr{S}$. Celle-ci permet de mesurer la portion de l'espace objectif qui est dominée par les solutions de l'ensemble considéré. La métrique $\mathscr{S}$ calcule un certain volume pour un ensemble de n solutions. Pour ce faire, pour chaque solution i, un polytope p_i est formé. Un polytope est construit par l'intersection de M (nombre de fonctions objectif) hyperplans. Pour chaque solution i et chaque axe f, il existe un unique hyperplan qui passe par i et qui est perpendiculaire à f.

Afin de mieux comprendre le calcul de $\mathscr{S}$, un exemple pour un problème de maximisation à deux objectifs est introduit. La Figure 1.5 présente l'espace objectif contenant les solutions de l'ensemble à évaluer. Dans un problème bi-objectif, les hyperplans sont des segments, les polytopes sont des rectangles et le volume à calculer est l'aire de l'ensemble des rectangles. Dans l'exemple, les deux hyperplans de $x_1 = (f_1(x_1), f_2(x_1))$ sont les segments $[f_1(x_1), x_1]$ et $[f_2(x_1), x_1]$. De plus, le polytope p_1 associé à la solution x_1 est le rectangle $(0, f_1(x_1), x1, f_2(x_1))$. La métrique d'hyper-volume $\mathscr{S}$ de l'ensemble des solutions correspond à l'aire de la forme colorée sur la Figure 1.5 qui est l'union de tous les polytopes.

$\mathscr{S}$ permet de quantifier la couverture de l'espace objectif d'un ensemble considéré. De plus grandes valeurs sont préférables, mais ne permettent pas de définir si elles sont obtenues majoritairement par la convergence ou par la diversité des solutions. C'est pour cela que cette métrique ne peut être utilisée toute seule pour évaluer les performances d'un algorithme multi-objectifs.

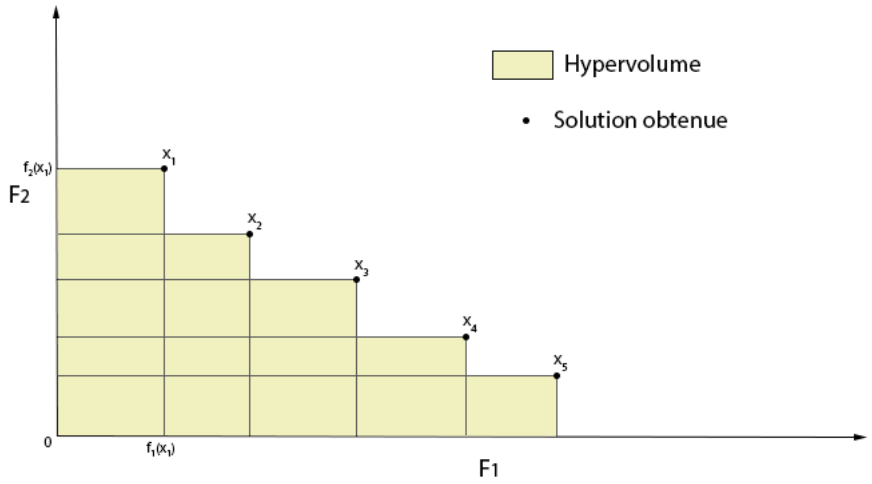


Figure 1.5: Calcul de la métrique $\mathscr{S}$ sur un problème de maximisation à deux objectifs

1.2.3 LA COUVERTURE DE DEUX ENSEMBLES : MÉTRIQUE $\mathscr{C}$

Une métrique binaire permet de comparer directement deux approches de résolution et plus spécifiquement leur ensemble de solutions obtenues. Ce type de métrique permet de répondre à certaines limitations des métriques unaires, notamment sur leur généricité, et constitue une alternative intéressante (Zitzler *et al.*, 2002b). La métrique de couverture de deux ensembles de solutions, également appelée métrique $\mathscr{C}$, a été proposée par Zitzler (1999). $\mathscr{C}(A,B)$ correspond à la proportion de solutions de l'ensemble B dominées par au moins une solution de l'ensemble A. La formulation de la métrique $\mathscr{C}$ est indiquée par l'Équation. 1.5

$$\mathscr{C}(A,B) = \frac{|b \in B | \exists a \in A : a \succ b|}{|B|} \tag{1.5}$$

À noter que si $\mathscr{C}(A,B) = 1$, alors toutes les solutions de B sont dominées par des solutions de A. Au contraire, si $\mathscr{C}(A,B) = 0$ alors aucune solution de A ne domine de solutions de B.

Lorsque les valeurs de métriques unaires sont très proches pour deux ensembles de solutions, la métrique de couverture $\mathscr{C}$ permet de différencier explicitement leur convergence. Généralement, lors d'une phase d'expérimentation, le calcul de $\mathscr{C}(A,B)$ et $\mathscr{C}(B,A)$ sera effectué afin de mieux comparer les ensembles A et B.

1.2.4 VALEUR D'ESPACEMENT MINIMAL : ms

Pour mesurer la diversité d'un ensemble de solutions A, Bandyopadhyay *et al.* (2004) proposent de calculer une valeur d'espacement minimal (*minimal spacing*) qui se base sur les distances entre les solutions pour chaque fonction objectif. En effet, pour chaque solution, on calcule la distance à sa plus proche voisine.

La valeur d'espacement minimal *ms* est définie comme suit :

$$ms = \sqrt{\frac{1}{|A|-1}\sum_{i=1}^{|A|-1}(d_i - d)^2} \tag{1.6}$$

avec $d_i = min_{j \in A} \sum_{m=1}^{M} |f_m^i - f_m^j|$, où M est le nombre de fonctions objectif, f_m^i est la valeur de la fonction objectif f pour la solution i et d est la moyenne des d_i. Une valeur de *ms* proche de 0 indique une bonne répartition des solutions dans l'espace objectif. En prenant en compte seulement les voisins qui n'ont pas encore été considérés, cette métrique pénalise les ensembles qui présentent des grands trous dans l'espace objectif.

1.3 CLASSIFICATION CONCEPTEUR DES MÉTHODES DE RÉSOLUTION MULTI-OBJECTIFS

Les méthodes de résolution peuvent être catégorisées selon la classification concepteur proposée par Talbi (2000). Cette classification distingue les méthodes selon leur manière de résoudre les problèmes multi-objectifs. La première catégorie regroupe les approches qui transforment le problème multi-objectifs en un problème uni-objectif, plus classique à résoudre. La deuxième catégorie concerne les approches qui traitent séparément les différentes fonctions objectif sans utiliser les notions de dominance au sens Pareto. Enfin, les approches Pareto utilisent directement les concepts de Pareto définis précédemment dans leur processus de recherche. Dans la suite de ce document, les principales approches de résolution de problèmes multi-objectifs sont présentées selon la classification concepteur.

1.3.1 MÉTHODES DE TRANSFORMATION VERS L'UNI-OBJECTIF

Les approches de transformation vers l'uni-objectif combinent les différentes fonctions objectif afin d'en former une seule. Une fois le problème devenu uni-objectif, il peut être résolu par les méthodes d'optimisation classiques, c'est-à-dire aussi bien des méthodes exactes que des heuristiques (algorithme génétique, algorithme de colonie de fourmis, etc.).

Ces méthodes permettent généralement de trouver une seule solution par exécution. Elles sont efficaces lorsque le décideur est clairement capable de définir ses préférences, de pondérer les objectifs ou d'identifier les buts à atteindre (approches a priori). Par ailleurs, lorsque le décideur n'a pas assez de connaissances à propos du problème, ces méthodes permettent de trouver des ensembles de solutions en faisant varier les facteurs utilisés (p. ex. pondérations, préférences). Le décideur peut ensuite choisir parmi les solutions proposées (approches a posteriori).

Différentes méthodes qui permettent la transformation d'un problème multi-objectifs en un problème uni-objectif sont présentées brièvement par la suite. Tout d'abord, la méthode de la somme pondérée peut être considérée comme la plus simple à mettre en place (Berro, 2001). Les différentes fonctions objectif sont additionnées pour n'en former qu'une seule. Des poids sont attribués à chaque fonction objectif afin de mesurer leur importance. L'Équation 1.7 formalise la transformation d'un problème multi-objectifs en un problème uni-objectif selon la somme pondérée.

$$\begin{aligned} \min \quad & F(x) = \sum_{m=1}^{M} w_m f_m(x) \\ \text{s.c} \quad & x \in X \end{aligned} \tag{1.7}$$

où $F(x)$ est la nouvelle fonction objectif à minimiser, M est le nombre de fonctions objectif initiales et f_m dénote la m-ième fonction objectif. Enfin, $w_m \in [0,1]$ est le poids de la fonction objectif f_m et la somme des poids est généralement égale à 1.

Pour éviter à l'utilisateur de pondérer les objectifs, plusieurs combinaisons de poids peuvent être testées afin d'obtenir plusieurs solutions à proposer au décideur. D'après Miettinen (1999), si le front Pareto est convexe, alors toutes les solutions Pareto optimales peuvent être obtenues

par cette méthode. Cependant, si le front est non-convexe alors il n'est pas garanti de trouver une solution Pareto optimale.

La méthode ε-contrainte a été proposée par Haimes *et al.* (1971) dans le but de pallier aux inconvénients de la méthode de la somme pondérée pour les fronts non-convexes. Selon cette méthode, il faut minimiser une seule fonction objectif et faire apparaître les autres fonctions objectif en tant que contraintes. L'Équation 1.8 décrit cette transformation.

$$\begin{aligned} \min \quad & f_k(x) \\ \text{s.c} \quad & f_m(x) \leq \varepsilon_m \quad m = 1,2...M \text{ et } m \neq k \\ & x \in X \end{aligned} \tag{1.8}$$

avec f_k une fonction objectif choisie, f_m la m-ième fonction objectif, M le nombre de fonctions objectif et ε_m la limite supérieure de f_m.

Cette méthode, comme la précédente, demande au décideur de définir certaines valeurs pour chaque fonction objectif. Néanmoins, cette fois-ci, les problèmes qui engendrent des fronts Pareto non-convexes sont solvables par cette méthode (Deb, 2001).

Une autre manière d'utiliser les pondérations des fonctions objectif est nommée méthode de la métrique pondérée (*Weighted Metric Method*) (Deb, 2001). Celle-ci nécessite le calcul du point Idéal z^*. Pour ce faire, chaque fonction objectif doit être minimisée de façon indépendante. La transformation du problème multi-objectifs peut alors se faire en sommant les distances des fonctions objectif par rapport au point Idéal. L'Équation 1.9 présente l'unique fonction objectif l_p à minimiser.

$$\begin{aligned} \min \quad & l_p(x) = \left(\sum_{m=1}^{M} w_m |f_m(x) - z_m^*|^p \right)^{1/p} \\ \text{s.c} \quad & x \in X \end{aligned} \tag{1.9}$$

avec f_m la m-ième fonction objectif et p un paramètre à determiner compris entre 1 et ∞. Si $p = \infty$, alors le problème se nomme problème de pondération de Tchebycheff et est décrit par l'Équation 1.10.

$$\begin{aligned} \min \quad & l_\infty(x) = \max_{m=1}^{M} w_m |f_m(x) - z_m^*| \\ \text{s.c} \quad & x \in X \end{aligned} \tag{1.10}$$

La programmation par but pour l'optimisation multi-objectifs a été proposée par Ignizio (1978). Une méthode de ce type vise à trouver une solution qui se rapproche le plus possible des attentes de l'utilisateur. Pour ce faire, l'utilisateur doit définir les buts qu'il souhaite atteindre. Une fois ses choix définis, il faut trouver la meilleure solution qui minimise la déviation par rapport à la solution attendue. Afin de réaliser cela, la formule utilisée par la méthode de la métrique pondérée (Équation 1.9) peut être réutilisée en changeant le point Idéal par la solution désirée (but à atteindre).

1.3.2 MÉTHODES NON-PARETO

Les méthodes non-Pareto n'utilisent pas les concepts de Pareto dans leur résolution. En effet, les algorithmes de cette catégorie considèrent les fonctions objectif indépendamment les unes des autres pendant leur déroulement. Le principal défaut de ces méthodes est qu'elles sont généralement incapables de trouver certaines parties du front Pareto (Coello, 2001). Deux algorithmes génétiques non-Pareto sont brièvement présentés par la suite.

L'algorithme VEGA (*Vector Evaluated Genetic Algorithm*), introduit par Schaffer (1985), est l'algorithme précurseur concernant les algorithmes génétiques multi-objectifs. En effet, c'est le premier algorithme génétique qui permet de récupérer un ensemble de solutions non-dominées et qui est donc une méthode a posteriori. Considérons un problème multi-objectifs à M fonctions objectif. Dans l'algorithme VEGA, à chaque génération, M sous-populations sont générées. Une sous-population p_i ($i = 1...M$) est obtenue en procédant à une phase de sélection classique dans laquelle la *fitness* correspond à la valeur de la fonction objectif f_i. Pour obtenir la nouvelle population, les sous-populations enfants sont combinées et les opérateurs de croisement et de mutation peuvent alors avoir lieu. Les résultats obtenus montrent un problème de diversité des solutions obtenues malgré le fait que l'opérateur de croisement s'effectue sur des individus pouvant se trouver dans des sous-populations différentes.

En 1996, Lis et Eiben (1996) proposent l'algorithme génétique MSGA (M*ulti-Sexual Genetic Algorithm*). Chaque individu d'une population est associé à un sexe qui correspond à l'indice de la fonction objectif par laquelle il est évalué. L'opérateur de croisement est effectué seulement sur des individus ayant un sexe différent et le nombre d'individus parents correspond au nombre de fonctions objectif. Ce croisement permet à une solution d'être performante dans plusieurs objectifs différents. Le sexe de l'enfant est le même que celui du parent qui a le plus de correspondance avec lui.

1.3.3 MÉTHODES PARETO

Les approches Pareto utilisent directement les concepts de dominance de Pareto, décrits à la Section 1.1.2, pour optimiser des problèmes multi-objectifs. Ce type d'approche a initialement été proposé par Goldberg (1989) pour les algorithmes génétiques. La dominance au sens Pareto permet généralement aux algorithmes de comparer les solutions entre elles.

Certaines métaheuristiques basées sur un individu unique, comme le recuit simulé (Bandyopadhyay *et al.*, 2008) ou la recherche avec tabous (Jaeggi *et al.*, 2008), ont été adaptées pour résoudre des problèmes multi-objectifs. De même, des algorithmes de recherche locale basés sur la dominance Pareto ont été proposés et ont connu un intérêt certain ces dernières années Dubois-Lacoste *et al.* (2015), notamment dans le but d'améliorer un ensemble de solutions obtenues par une autre métaheuristique.

Plusieurs travaux ont montré que les métaheuristiques basées sur une population sont particulièrement adaptées pour résoudre des problèmes multi-objectifs (Zhou *et al.*, 2011). En effet, en faisant évoluer une population de solutions, elles sont capables d'approximer un front Pareto en une seule exécution. Bien que les premiers algorithmes multi-objectifs Pareto proposés dans la littérature étaient non-élitistes (Fonseca et Fleming, 1993) (Horn *et al.*, 1994), différentes publications (Deb et Goel, 2001) (Zitzler, 1999) ont montré l'intérêt de l'élitisme dans la convergence vers le front Pareto. De ce fait, plusieurs travaux récents ont été réalisés autour de métaheuristiques Pareto élitistes (Leite *et al.*, 2015), (Tian *et al.*, 2016), (Sanhueza *et al.*, 2017). Les algorithmes multi-objectifs élitistes ont un mécanisme qui permet de conserver les meilleurs individus tout au long du déroulement de l'algorithme soit par l'intermédiaire d'une archive de solutions non-dominées, soit directement dans la population qui évolue. De par les spécificités et nécessités de l'optimisation multi-objectifs, les techniques élitistes permettent également de maintenir une diversité des solutions (Berro, 2001).

Pour les raisons précédemment évoquées et leur efficacité éprouvée, le reste de cette thèse se consacre exclusivement aux métaheuristiques multi-objectifs Pareto élitistes. La prochaine section de ce chapitre présente les principales méthodes de résolution correspondant à ce type de métaheuristique. Les métaheuristiques basées sur une population ont été beaucoup investiguées par le passé pour répondre aux problèmes multi-objectifs. Trois des principales métaheuristiques sont présentées par la suite : les algorithmes évolutionnaires multi-objectifs (AEMO), qui sont les plus étudiés dans la littérature concernant l'optimisation multi-objectifs

(Luna et Alba, 2015), les algorithmes par essaim particulaire et enfin les algorithmes de colonie de fourmis. De plus, ces dernières années, un nouvel intérêt a été suscité par les algorithmes de recherche locale basés sur la dominance Pareto (Dubois-Lacoste *et al.*, 2015). Quelques algorithmes de ce type et certaines hybridations avec des algorithmes évolutionnaires seront détaillés par la suite. Finalement, pour répondre à des problèmes impliquant une multitude d'objectifs (plus de trois objectifs) des alternatives plus efficaces que les approches multi-objectifs reconnues ont été proposées récemment (Deb et Jain, 2014) et sont exposées dans la suite de ce chapitre.

1.4 ALGORITHMES ÉVOLUTIONNAIRES MULTI-OBJECTIFS (AEMO) PARETO ÉLITISTES

Les algorithmes évolutionnaires sont basés sur la théorie de l'évolution de Charles Darwin. Dans ceux-ci, un individu représente une solution du problème et une population est un ensemble d'individus. La population évolue à travers des générations par le biais d'opérations génétiques (sélection, croisement et mutation). Pour un algorithme génétique, qui est un type particulier d'algorithme évolutionnaire, c'est l'attribution de la *fitness* qui diffère principalement entre l'optimisation uni-objectif et l'optimisation multi-objectifs. Il faut trouver une autre manière de définir la *fitness*, car celle-ci ne peut plus être définie comme simplement équivalente à la valeur de fonction objectif. Le déroulement classique d'un algorithme génétique est présenté à la Figure 1.6.

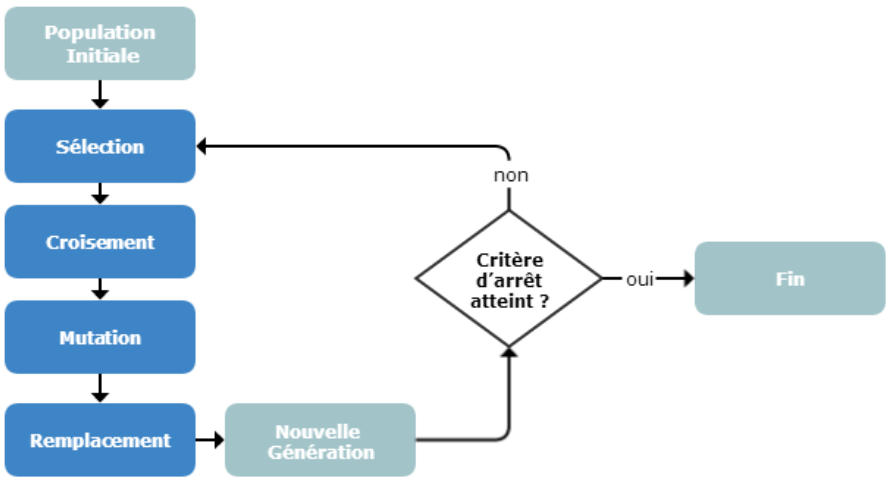


Figure 1.6: Schéma du déroulement d'un algorithme génétique

Plusieurs types d'AEMO Pareto élitistes ont été proposés dans la littérature et sont présentés dans les sections suivantes.

1.4.1 AEMO BASÉS SUR LA DOMINANCE PARETO

La première classe d'AEMO utilise principalement la notion de dominance Pareto pour comparer les individus entre eux. En outre, lors du remplacement élitiste, ces individus sont classés grâce à cette relation de dominance et seulement les meilleurs sont conservés.

1.4.1.1 COMPARAISON DE TROIS AEMO

Des algorithmes reconnus performants tels que SPEA-II (*Strength Pareto Evolutionary Algorithm II*) (Zitzler *et al.*, 2002a), NSGA-II (*Nondominated Sorting Genetic Algorithm II*) (Deb *et al.*, 2002) et GISMOO (*Genetic Immune Strategy for Multiple Objective Optimization)* (Zinflou *et al.*, 2012) utilisent deux facteurs pour qualifier les individus d'une population : un facteur de dominance et un facteur d'isolement. En plus de qualifier les solutions par ces deux facteurs, ces algorithmes ont d'autres caractéristiques communes et des particularités qui sont recensées au Tableau 1.1 et détaillées par la suite.

	Facteur de dominance	**Facteur d'isolement**	**Sélection**	**Remplacement**	**Archive**	**Particularité**
SPEA-II	Équation 1.11	Distance euclidienne	Tournoi binaire	Générationnel	Participative	Troncature de l'archive
NSGA-II	Tri par front	Crowding distance	Tournoi binaire	Élitiste	N/A	
GISMOO	Équation 1.12	Distance euclidienne	Tournoi binaire	Élitiste	Externe	Phase immunitaire

Tableau 1.1: Résumé des caractéristiques des algorithmes SPEA-II, NSGA-II et GISMOO

Facteur de dominance Le facteur de dominance mesure la convergence vers le front Pareto. Dans l'algorithme SPEA-II, il favorise les solutions les moins dominées (convergence), mais également les solutions dominées par des solutions dominant peu de solutions. Pour ce faire,

la notion de force d'un individu x, notée $S(x)$, est tout d'abord introduite. Elle correspond au nombre de solutions que domine x dans la population POP et l'archive A tel que $S(x) = |y|y \in POP \cup A, x \succ y|$. Le facteur de dominance R d'une solution est alors égal à la somme des forces des individus qui la dominent.

$$D(x) = \sum_{j \in Pop \cup A, j \prec i} S(j) \tag{1.11}$$

Les solutions non-dominées ont un facteur de dominance à 0 et les solutions dominées par beaucoup de solutions ont un facteur élevé.

Pour l'algorithme NSGA-II, le facteur de dominance est déterminé via une assignation de rang ou tri par front. Chaque rang ou front correspond à un niveau de dominance. Ainsi, les solutions de rang 1 (front 1) correspondent à des solutions non-dominées et les solutions de rang 2 sont des solutions dominées seulement par des solutions du front 1. Il en est de même pour les fronts suivants. Un exemple d'assignation de rang est donné à la Figure 1.7.

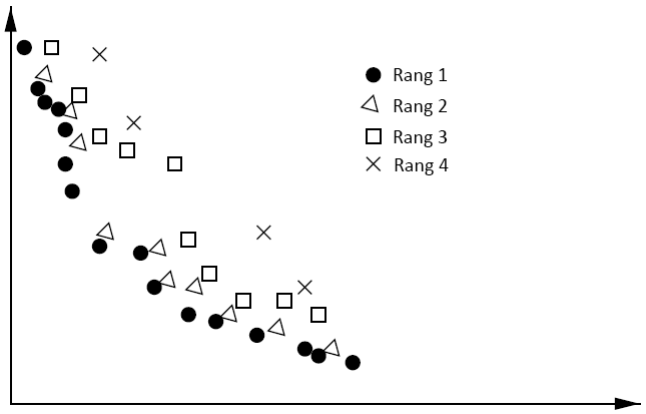


Figure 1.7: Exemple d'assignation de rang dans NSGA-II (Talbi, 2009)

Le facteur de dominance de GISMOO est similaire à celui utilisé dans SPEA-II. En reprenant $S(x)$, le nombre de solutions que domine x tel que $S(x) = |y|y \in POP \cup Q_t, x \succ y|$, le facteur de dominance $R^+(x)$ de GISMOO est défini par l'Équation 1.12.

$$R^+(x) = \begin{cases} \frac{S(x)}{1+2 \cdot S(x)} & \text{si } x \text{ est non-dominée} \\ \sum\limits_{y \in POP_t \cup Q_t, y \succ x} S(y) & \text{sinon} \end{cases} \tag{1.12}$$

Facteur d'isolement Le facteur d'isolement permet de favoriser les solutions les plus éloignées des autres et ainsi orienter la recherche vers des zones moins explorées. Dans les algorithmes SPEA-II et GISMOO, le facteur d'isolement d'une solution est égal à la distance euclidienne minimale à une autre solution. Dans l'algorithme NSGA-II, ce facteur est appelé *crowding distance* et est obtenu en calculant la moyenne des deux plus proches voisins de la solution pour chaque objectif.

Sélection et remplacement Le type de sélection utilisé par chacun des algorithmes est un tournoi binaire. Pour comparer deux solutions, le facteur de dominance est tout d'abord pris en compte. De ce fait, la solution avec le meilleur facteur de dominance est sélectionnée. En cas d'égalité, la solution avec le plus grand facteur d'isolement est privilégiée. Le remplacement effectué par SPEA-II est le remplacement générationnel. Selon ce dernier, les nouvelles générations sont composées uniquement des individus enfants et les individus parents sont supprimés. Au contraire, les phases de remplacement des algorithmes NSGA-II et GISMOO se veulent élitistes. Les meilleurs individus d'une génération à l'autre sont ainsi conservés dans la population Parent. Les meilleurs individus sont les solutions avec le meilleur facteur de dominance. Si plusieurs solutions ont le même facteur de dominance, leur facteur d'isolement est alors pris en compte.

Gestion d'une archive Parmi les trois algorithmes présentés, seul l'algorithme NSGA-II n'utilise pas d'archive pour conserver les solutions non-dominées. Dans l'algorithme GISMOO, l'archive est externe, car elle n'intervient pas directement dans les différentes opérations génétiques. Elle est mise à jour à chaque nouvelle solution créée. Au contraire, l'algorithme SPEA-II utilise une archive qui intervient dans l'assignation du facteur de dominance. L'archive utilisée dans SPEA-II, ayant une capacité limitée, doit être tronquée lorsque le nombre de solutions contenues est trop grand. Pour ce faire, la méthode de regroupement hiérarchique est utilisée (Morse, 1980).

Particularité de GISMOO La principale particularité de GISMOO est que la moitié de la population Enfant est générée par le biais d'une phase immunitaire empruntée aux systèmes immunitaires artificiels. Cette phase se fait en plusieurs étapes. Tout d'abord, il faut sélectionner les individus, nommés anticorps, qui seront clonés. Dans GISMOO, les anticorps correspondent aux solutions non-dominées de la population Parent. Ensuite, il faut déterminer, pour chaque

anticorps, combien de clones seront créés à partir de celui-ci. Ce nombre dépend du facteur d'isolement associé à la solution. Ainsi, plus une solution est isolée, plus des clones seront créés à partir de celle-ci. Ce mécanisme permet d'intensifier les recherches dans les zones moins peuplées et donc d'améliorer la diversité des solutions obtenues. Deux copies de chaque clone sont créées et mutées. Seul le meilleur clone est ajouté à la nouvelle population, c'est-à-dire celui qui domine l'autre. Le processus général de GISMOO est schématisé à la Figure 1.8.

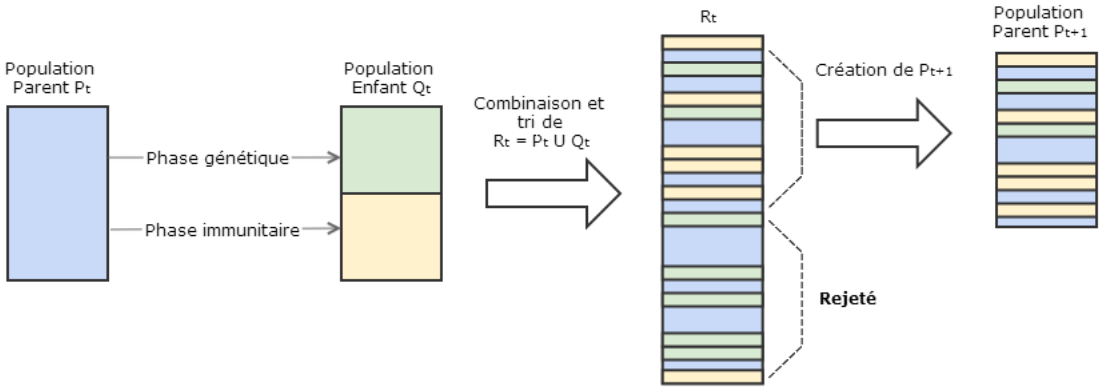


Figure 1.8: Schéma de fonctionnement de l'algorithme GISMOO (Zinflou *et al.*, 2012)

Performances L'algorithme NSGA-II reste aujourd'hui une référence dans le domaine des métaheuristiques multi-objectifs. En effet, certains travaux l'utilisent pour résoudre de nouveaux problèmes (Leite *et al.*, 2015), d'autres tentent de l'améliorer (Qiu *et al.*, 2015) et enfin les derniers l'utilisent comme base de comparaison (White et He, 2012). Cependant, l'algorithme GISMOO proposé plus récemment a été prouvé compétitif par rapport à NSGA-II et SPEA-II sur différents problèmes multi-objectifs (Zinflou *et al.*, 2012) (Zinflou *et al.*, 2013) (Gagné C., 2013).

1.4.1.2 AUTRES AEMO BASÉS SUR LA DOMINANCE PARETO

D'autres AEMO basés sur la dominance Pareto ont été proposés et présentent un intérêt pour leur mode de fonctionnement ou pour leur performance. L'algorithme PESA-II (Corne *et al.*, 2001) présente la particularité d'effectuer sa sélection sur une région (hyperbox) de l'espace objectif. Une sélection par tournoi est effectuée pour choisir de manière probabiliste

les régions contenant le moins de solutions. La détermination des individus qui participeront aux opérations génétiques se fait aléatoirement parmi les régions sélectionnées. Le découpage en régions requiert des calculs et des paramètres supplémentaires.

L'algorithme HSS-EA (Zhang *et al.*, 2014) reprend le principe de l'algorithme VEGA pour trouver des solutions proches du front Pareto, mais orientées vers un seul objectif. Pour trouver les solutions de compromis, les auteurs ajoutent une méthode d'assignation de *fitness* qui dépend du nombre de solutions dominées. Karasakal et Silav (2015) proposent une version modifiée de l'algorithme SPEA-II dans laquelle ils incorporent le *crowding distance* de NSGA-II.

1.4.2 AEMO BASÉS SUR UN INDICATEUR DE PERFORMANCE

Plutôt que d'utiliser la dominance Pareto dans leur mécanisme de sélection, une autre classe d'AEMO utilise un indicateur de performance (ou métrique). Zitzler et Künzli (2004) ont été les premiers à proposer une telle approche et proposèrent l'algorithme IBEA qui ne requiert pas de mécanisme de diversification particulier. Un indicateur de performance binaire, qui permet la comparaison de deux ensembles de solutions, est utilisé. La fitness d'un individu correspond à la perte de qualité (selon l'indicateur) engendrée par la suppression de cet individu au sein de la population. Ainsi, plus cette valeur est grande, meilleur est l'individu. Les phases de sélection et de remplacement sont ainsi basées sur la fitness des individus.

De la même manière, l'algorithme SMS-EMOA (Beume *et al.*, 2007) cherche à maximiser la métrique d'hypervolume à chaque génération. Pour ce faire, lors de la phase de remplacement, un tri par front est tout d'abord effectué comme dans NSGA-II (Deb *et al.*, 2002). Dans le dernier front sélectionné, les individus qui contribuent le moins à la valeur de l'hypervolume sont supprimés. Comme le calcul de la métrique d'hypervolume peut être excessivement long, Bader et Zitzler (2011) ont défini des méthodes pour accélérer ce calcul, notamment pour résoudre des problèmes avec plus de quatre objectifs.

Enfin, d'autres métriques que l'hypervolume peuvent être utilisées comme indicateur de performance. Brockhoff *et al.* (2015) ont ainsi étudié la possibilité d'utiliser une alternative plus rapide à calculer, la métrique $R2$ (Hansen et Jaszkiewicz, 1998), dans un AEMO qu'ils ont nommé R2-EMOA. Menchaca-Mendez et Coello (2015) ont incorporé la métrique de convergence *GD* au sein de leur algorithme GDE-MOEA. Enfin, une version améliorée de

l'indicateur *IGD* a été définie par Tian *et al.* (2016) et est utilisée comme base pour comparer des individus.

1.4.3 AEMO BASÉS SUR LA DÉCOMPOSITION

La dernière classe d'AEMO concerne les approches qui décomposent le problème multi-objectifs en un certain nombre de problèmes uni-objectif en s'appuyant généralement sur les algorithmes de transformation vers l'uni-objectif décrits à la Section 1.3.1. Bien que les prémisses de cette approche soient décrites par Ishibuchi et Murata (1998), c'est l'algorithme MOEA/D proposé par Qingfu et Hui (2007) qui a popularisé, de par ses performances, cette classe d'AEMO. Dans MOEA/D, la méthode de Tchebycheff est, entre autres, utilisée pour subdiviser le problème original en sous-problèmes. Un algorithme évolutionnaire classique permet de faire évoluer une population dans laquelle chaque individu correspond à la meilleure solution d'un sous-problème spécifique trouvée depuis le début de l'algorithme. Une relation de voisinage, basée sur les distances entre les vecteurs poids, est définie entre les sous-problèmes. Pour la phase sélection et pour chaque sous-problème i, deux individus sont sélectionnés parmi le voisinage de i pour participer aux croisements. De même, pendant l'opération de remplacement, les enfants générés par les individus voisins de i peuvent prendre la place de l'individu i.

Basés sur les grandes lignes MOEA/D, de nombreux algorithmes ont été proposés pour l'améliorer Trivedi *et al.* (2017). Certains travaux se focalisent sur l'optimisation de la génération des vecteurs poids utilisés pour la décomposition (Giagkiozis *et al.*, 2014). D'autres s'intéressent plus particulièrement à la méthode de décomposition en elle-même (Sato, 2015) (Wang *et al.*, 2016). Enfin, les derniers proposent des méthodes pour améliorer la phase de sélection ou de remplacement (Li *et al.*, 2014) (Li *et al.*, 2015).

1.5 AUTRES MÉTAHEURISTIQUES PARETO ÉLITISTES

1.5.1 ALGORITHMES PAR ESSAIM PARTICULAIRE

L'algorithme par essaim particulaire, proposé par Eberhart et Kennedy (1995), fait évoluer une population (ou essaim) de particules qui se déplace sur l'espace de recherche afin de trouver les meilleures solutions possibles. Les déplacements des particules s'effectuent selon

une vitesse, la meilleure position trouvée par la particule et la meilleure position de l'essaim. La principale différence entre la résolution des problèmes multi-objectifs et uni-objectif réside dans la détermination de la meilleure position trouvée par l'essaim. L'algorithme MOPSO (Multiple Objective Particle Swarm Optimization) (Coello et Lechuga, 2002) est la première adaptation de l'optimisation par essaim particulaire pour des problèmes multi-objectifs qui utilise les notions de dominance Pareto. Deux archives sont maintenues. La première archive permet d'enregistrer l'ensemble des solutions non-dominées, l'archive globale. La seconde est l'archive locale, propre à chaque particule, qui sauvegarde sa meilleure solution trouvée jusqu'alors. La solution servant de guide pour la mise à jour de la vitesse est sélectionnée dans l'archive globale. Cette sélection se fait par un facteur d'isolement, similaire à PESA-II, qui garantit une bonne diversité. Par ailleurs, l'archive globale a une certaine limite de capacité et si celle-ci est dépassée, les solutions les moins isolées dans l'espace de recherche sont supprimées.

L'algorithme TV-MOPSO (Tripathi *et al.*, 2007) reprend les méthodes du précédent algorithme et ajoute des coefficients qui permettent de contrôler les variations de vitesse. Pour ce faire, au début de l'algorithme, l'exploration est privilégiée, puis au fil des itérations les coefficients sont modifiés afin de favoriser de plus en plus l'exploitation. Dans l'algorithme DSMOPSO (Li *et al.*, 2010), la solution de l'archive qui sert de guide est choisie grâce au *crowding distance* utilisé dans NSGA-II. L'algorithme NSPSO (Li, 2003) considère les meilleures solutions de chaque particule et les nouvelles solutions générées comme une unique population et une méthode de remplacement similaire à celle de NSGA-II est utilisée afin de conserver les meilleures solutions à chaque itération. Enfin, Dai *et al.* (2015) proposent l'algorithme MPSO/D qui décompose l'espace objectif en sous-régions et fait en sorte que chacune d'entre elles soit occupée par une particule.

1.5.2 ALGORITHMES DE COLONIE DE FOURMIS

L'algorithme de colonie de fourmis a été également investigué pour résoudre des problèmes multi-objectifs. Dans un tel algorithme, des traces de phéromones sont associées aux objets du problème et influent les fourmis dans leur construction de solution. Ainsi, les objets appartenant aux meilleures solutions recevront plus de phéromones que les autres. Comme le révèle la taxonomie de (García-Martínez *et al.*, 2007), les algorithmes de colonies de fourmis multi-objectifs peuvent être classés selon le nombre de matrices de phéromones utilisées par

colonie. Certains algorithmes (Baran et Schaerer, 2003) (Angus, 2007) utilisent une seule matrice qui doit représenter l'ensemble des objectifs. Dans d'autres algorithmes (Cardoso *et al.*, 2003)(Doerner *et al.*, 2004), plusieurs matrices de phéromones sont utilisées et chacune d'entre elles correspond à un objectif spécifique. Pour exploiter ces matrices, il convient soit d'utiliser une méthode d'agrégation (Cardoso *et al.*, 2003) pour chaque fourmi ou soit d'associer un sous-ensemble de fourmis à une unique matrice de phéromones (Iredi *et al.*, 2001).

Dans une étude effectuée par Özkale et Fığlalı (2013), douze algorithmes de colonie de fourmis multi-objectifs sont empiriquement comparés dans la résolution du problème d'assignation quadratique multi-objectifs. Basés sur les principes de décomposition, Ke *et al.* (2013) ont proposé une hybridation de l'algorithme MOEA/D (Qingfu et Hui, 2007) avec un algorithme de colonie de fourmis. Chaque fourmi est responsable de l'optimisation d'un sous-problème.

1.5.3 ALGORITHMES DE RECHERCHE LOCALE ET ALGORITHMES MÉMÉTIQUES

Des algorithmes de recherche locale basés sur la dominance au sens Pareto, que l'on nommera par la suite algorithmes PLS (*Pareto Local Search*), ont été également proposés pour répondre à des problèmes multi-objectifs. Ces algorithmes peuvent être utilisés pour améliorer un ensemble de solutions obtenues par une autre métaheuristique (Lust et Teghem, 2010a) (Lust et Teghem, 2010c). Alors que la recherche locale en uni-objectif travaille continuellement sur une seule solution, les algorithmes PLS s'occupent d'optimiser un ensemble de solutions appelé archive.

Paquete *et al.* (2004) ont proposé le premier algorithme PLS dont les grands lignes sont présentés à L'Algorithme 1. Dans celui-ci, un ensemble de solutions initiales contenu dans une archive A_0 est amélioré par le biais de recherche locale. Une autre archive A permet de conserver l'ensemble des solutions non-dominées et est retournée par l'algorithme. Pour ce faire, un voisinage est calculé pour toutes les solutions $s \in A_0$ (lignes 4-9). Chaque voisin s' de s qui n'est pas dominé par une solution de l'archive A est ajouté à A (lignes 6-8). Une fois que le voisinage de s est exploré, la solution est marquée comme visitée pour ne pas être explorée de nouveau (ligne 10). A_0 est mise à jour à chaque itération et contient l'ensemble des solutions non explorées (ligne 11). L'algorithme s'arrête quand toutes les solutions de l'archive ont été explorées.

Algorithme 1: Algorithme de recherche locale Pareto (Paquete *et al.*, 2004)

```
1  Initialisation de l'archive A_0;
2  Archive A = A_0;
3  Tant que A_0 n'est pas vide Faire
4  |  Choisir une solution s ∈ A_0;
5  |  Pour chaque voisin s' de s Faire
6  |  |  si r ⊀ s', c∀r ∈ A alors
7  |  |  |  Mettre à jour A avec s';
8  |  |  finsi
9  |  FinPour
10 |  Marquer s comme explorée;
11 |  A_0 = {s ∈ A | s est non explorée} ;
12 FinTantque
13 retourner A;
```

Variantes de l'algorithme Alsheddy et Tsang (2010) proposent un algorithme basé sur la PLS combinée à une recherche locale guidée (*Guided Local Search*). Cette dernière est une méthode qui s'ajoute à une recherche locale dans le but de sortir des minimums locaux. Ainsi, quand l'algorithme reste bloqué dans un minimum local, une fonction de pénalité est utilisée pour dégrader sa fonction objectif afin d'orienter la recherche vers d'autres solutions. L'algorithme Two-phase Pareto Local Search (2PPLS) (Lust et Teghem, 2010c), comme son nom l'indique, est un algorithme de recherche locale composé de deux phases pour générer un ensemble de solutions non-dominées proches du front Pareto. La première phase permet d'obtenir les solutions supportées, c'est-à-dire, les solutions qui peuvent être découvertes par des méthodes d'agrégation comme la somme pondérée. La deuxième phase permet d'obtenir des solutions optimales non supportées en appliquant la PLS sur l'ensemble de solutions obtenues durant la première phase. Dans une étude récente, Dubois-Lacoste *et al.* (2015) analysent et améliorent le comportement *anytime* des algorithmes PLS. Le comportement *anytime* correspond à la capacité d'un algorithme à obtenir des solutions de bonne qualité à n'importe quel moment de l'exécution de l'algorithme. Comme le soulignent les auteurs, l'algorithme PLS de base n'est pas un algorithme *anytime*, car il ne permet pas de trouver des bonnes solutions rapidement. Pour améliorer l'algorithme original, différentes options sont proposées, notamment sur les stratégies de voisinage.

Algorithmes mémétiques Des algorithmes mémétiques ont également été conçus pour répondre à des problèmes multi-objectifs. Ces algorithmes hybrident un algorithme génétique avec une recherche locale qui permet d'obtenir une meilleure convergence vers le front Pareto. Dans les algorithmes décrits par Adra *et al.* (2009) et Delgado *et al.* (2008), les meilleures solutions trouvées à chaque génération sont améliorées par une recherche locale. Par ailleurs, Caponio et Neri (2009) hybrident l'algorithme NSGA-II avec deux types de recherche locale. Enfin, l'algorithme PMS^{mo} (Zinflou *et al.*, 2008) utilise deux archives pour maintenir des solutions non-dominées et attribue des facteurs de dominance et d'isolement de manière similaire à GISMOO.

1.5.4 ALGORITHMES MANY-OBJECTIVES

Bien que des algorithmes basés sur la dominance au sens Pareto, tels que NSGA-II, ont été prouvés performants pour des problèmes à deux ou trois objectifs, leur difficulté à répondre à des problèmes impliquant plus d'objectifs a été démontrée dans quelques travaux (Hughes, 2005), (Khare *et al.*, 2003). Plusieurs raisons expliquent cette difficulté. D'une part, comme le nombre de solutions non-dominées d'une population augmente exponentiellement avec le nombre d'objectifs, la différenciation des meilleures solutions, notamment dans les phases de sélection, est plus difficile. Ainsi, le processus de recherche est dégradé puisque la sélection se fait presque aléatoirement (Garza-Fabre *et al.*, 2009). D'autre part, les méthodes de préservation de diversité, comme le *crowding distance* de NSGA-II, ont des coûts de calcul trop importants (Deb et Jain, 2014) et ont besoin d'être adaptées.

Certains problèmes réels, nommés problèmes *many-objectives*, impliquent par définition de nombreux objectifs (Coello et Lamont, 2004) (Sülflow *et al.*, 2007). Comme les algorithmes multi-objectifs reconnus sont moins performants pour les résoudre, depuis quelques années, une attention particulière est portée à ces problèmes. Dans la littérature, les auteurs s'accordent à dire qu'il s'agit de problèmes composés d'au moins trois objectifs.

L'algorithme NSGA-III, proposé par Deb et Jain (2014), reprend les grandes lignes de l'algorithme NSGA-II. L'assignation de rang est conservée, mais le facteur d'isolement (*crowding distance*) n'est plus utilisé. Comme dans NSGA-II, seuls les premiers fronts sont ajoutés à la nouvelle population. Lorsqu'un front F_l ne peut pas être ajouté entièrement, il faut faire une sélection de solutions dans le but de maintenir une certaine diversité. Dans l'algorithme NSGA-III, des points de référence préalablement calculés ou définis par l'utilisateur sont

utilisés. Diverses techniques permettent d'obtenir des points de référence bien espacés sur l'espace objectif (Das et Dennis, 1998). Un exemple de quinze points de référence pour un problème à trois objectifs est donné à la Figure 1.9.

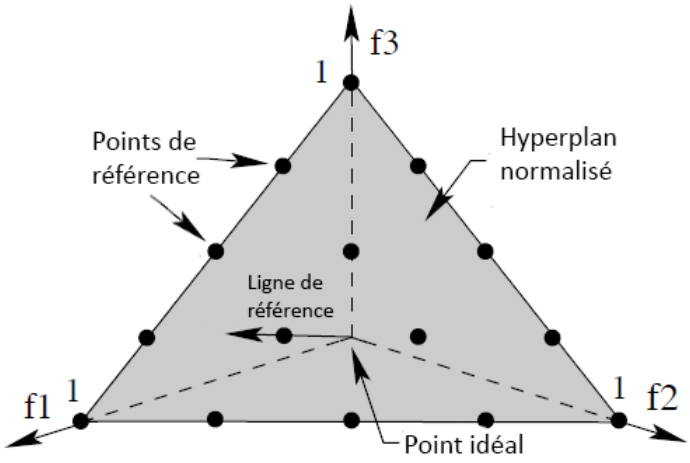


Figure 1.9: Exemple de points de référence pour un problème à trois objectifs (Deb et Jain, 2014)

Après une normalisation des objectifs, chaque solution de la population est associée au plus proche point de référence. Pour ce faire, les distances perpendiculaires des solutions aux lignes formées par les points de référence et le point d'origine sont calculées. Les solutions du front F_l qui sont ajoutées à la nouvelle population sont celles associées à des points de référence moins représentés (les moins associés à des solutions). Cela permet de privilégier des solutions qui se situent dans des zones de l'espace objectif moins explorées. L'algorithme NSGA-III a été prouvé efficace pour des instances de problèmes continus impliquant de trois à quinze objectifs (Deb et Jain, 2014).

Une version améliorée de NSGA-III, nommée θ-NSGA-III, a été proposée par Yuan *et al.* (2014) et a pour but d'améliorer la convergence de l'algorithme original. Le tri par front n'est cette fois-ci pas réalisé avec la dominance au sens Pareto, mais avec une nouvelle forme de dominance : la θ-dominance. Cette dernière prend à la fois en compte la convergence d'une solution vers le point Idéal, mais aussi la diversité. Finalement, selon l'expérimentation réalisée par les auteurs, l'algorithme θ-NSGA-III surclasse NSGA-III sur des problèmes continus DTLZ de quatre à vingt objectifs. Enfin, une version générique de NSGA-III, nommée U-NSGA-III (Seada et Deb, 2015), permet de résoudre des problèmes mono-objectif, multi-objectifs ou même *many-objectives*.

1.6 CONCLUSION

Au cours de ce chapitre, les particularités et caractéristiques de l'optimisation multi-objectifs ont tout d'abord été abordées. Ainsi, les difficultés qu'amène la résolution d'un problème à plusieurs objectifs ont été identifiées. Il a été vu qu'il existe de nombreuses approches pour résoudre de tels problèmes. Ces approches présentent des caractéristiques et des comportements communs, par exemple la gestion d'une archive de solutions non-dominées, l'utilisation d'une population, l'aspect élitiste des méthodes ou encore l'attribution de facteurs de dominance et d'isolement pour certains algorithmes évolutionnaires. De plus, la majorité des métaheuristiques, utilisant les notions de dominance Pareto, permettent d'obtenir des solutions proches du front Pareto en un temps réduit. Cependant, ce temps peut demeurer excessivement long pour des instances de problèmes de grande taille (Luna *et al.*, 2006). Par exemple, la résolution approchée d'un problème de voyageur de commerce multi-objectifs contenant un millier de villes requiert quelques heures de calcul (Lust et Jaszkiewicz, 2010). De même, certaines méthodes, comme le tri par front ou l'assignation d'un facteur d'isolement pour chaque solution, présentent une complexité élevée ce qui augmente le temps d'exécution. La gestion d'une archive demande également de nombreuses ressources de calcul, mais aussi des espaces mémoire importants lorsque l'archive n'a pas une capacité limitée. De ce fait, dans le but de diminuer le temps d'exécution des algorithmes multi-objectifs et de pouvoir résoudre des problèmes de grande taille en un temps qualifié d'acceptable, de nouvelles méthodes doivent être employées (Luna *et al.*, 2006). Les nouvelles architectures de calcul haute performance deviennent, dans ce contexte, des outils puissants pour mettre en place des stratégies parallèles qui permettent la résolution efficace de problèmes à grande échelle (Talbi, 2018). Ces stratégies font l'objet du prochain chapitre.

CHAPITRE 2

EXPLOITATION DES ARCHITECTURES DE CALCUL HAUTE PERFORMANCE POUR AMÉLIORER LES MÉTAHEURISTIQUES MULTI-OBJECTIFS

Au cours des années 2000, la puissance des cœurs de processeur CPU a faiblement évolué et un plafonnement de la fréquence d'horloge a été observé. Pour continuer d'augmenter significativement la puissance des processeurs, les constructeurs ont multiplié le nombre de cœurs par puce si bien qu'à partir de 2009, les ordinateurs standards sont dans la grande majorité constitués de processeurs à deux ou quatre cœurs (Rauber et Rünger, 2010). De même, ce type de processeur a contribué à l'augmentation considérable de la puissance des architectures de calcul haute performance qui sont de plus en plus efficaces et accessibles de nos jours. Afin d'exploiter au maximum le potentiel de ces machines parallèles et obtenir des accélérations importantes, une maîtrise de l'environnement matériel et logiciel est nécessaire. En outre, le calcul à haute performance a permis de nombreuses avancées dans des domaines variés tels que la chimie, la physique, les mathématiques, etc. Il est donc naturel d'identifier ce que les machines parallèles peuvent apporter au domaine de la recherche opérationnelle et plus particulièrement à l'optimisation multi-objectifs. En effet, la résolution d'un problème multi-objectifs réel requiert généralement des capacités de calcul importantes. Même si des métaheuristiques multi-objectifs permettent d'optimiser les problèmes en un temps diminué, certaines améliorations sont encore possibles et peuvent être amenées par le parallélisme. Premièrement, le temps d'exécution peut être encore réduit, notamment pour des problèmes complexes qui sont associés à des centaines voire des milliers de variables de décision. Ce temps est un aspect important dans un monde de plus en plus concurrentiel. Deuxièmement, puisqu'il s'agit de méthodes approchées, il peut être encore possible d'améliorer la qualité des solutions. Dans le cadre de l'optimisation multi-objectifs, la convergence vers le front Pareto et la distribution des solutions sont visées. Finalement, le parallélisme peut permettre de

manipuler des instances de problèmes de grandes tailles qui requièrent des espaces mémoire importants.

Dans ce chapitre, après avoir fait une introduction au parallélisme, un état de l'art des approches de parallélisation existantes pour les métaheuristiques multi-objectifs est effectué avec une attention plus particulière aux algorithmes évolutionnaires qui constituent une majeure partie des travaux dans ce domaine. Cette revue de littérature comprend des méthodes s'appliquant à des problèmes combinatoires et qui exploitent différents types d'architecture. Les objectifs de la recherche sont finalement exposés dans la dernière partie du chapitre.

2.1 INTRODUCTION AU PARALLÉLISME

Depuis plusieurs dizaines d'années, les limites des calculs par le biais d'un unique processeur sont apparues comme un important problème pour les scientifiques. Ces limites sont rencontrées à la fois au niveau des temps de calcul, mais aussi au niveau des espaces mémoire nécessaires. Pour accélérer le temps d'exécution d'un programme, les chercheurs se sont intéressés à la possibilité de faire des opérations sur plusieurs processeurs à la fois. Les recherches effectuées ont donné naissance au *parallélisme* qui consiste à implémenter sur une architecture parallèle un programme exécutant simultanément plusieurs tâches. Le processus qui permet de passer d'un programme séquentiel à un programme parallèle est le processus de *parallélisation*. L'implémentation d'un algorithme parallèle dépend fortement de l'architecture matérielle sur lequel sera exécuté le programme, mais elle est également influencée par l'environnement logiciel. Ce dernier comprend le système d'exploitation, le langage de programmation ainsi que le compilateur. La première partie de cette section se consacre à la description des principales architectures système. Ensuite, les environnements logiciels associés à ces architectures sont introduits. Finalement, un modèle de parallélisation et les métriques permettant de mesurer la qualité d'une parallélisation sont présentés.

2.1.1 ENVIRONNEMENT MATÉRIEL

Dans les années soixante-dix, Flynn (1972) propose une manière de classifier les architectures parallèles : c'est la taxonomie de Flynn. Elle permet de différencier les systèmes selon les caractéristiques de leurs flux de données et de leurs flux d'instructions. Afin de mettre à jour cette taxonomie et de distinguer les nouvelles architectures, (Tanenbaum, 2005) propose

de subdiviser la catégorie des systèmes **MIMD** (Multiple Instruction, Multiple Data) en deux principales classes. Cette nouvelle taxonomie est proposée à la Figure 2.1 .

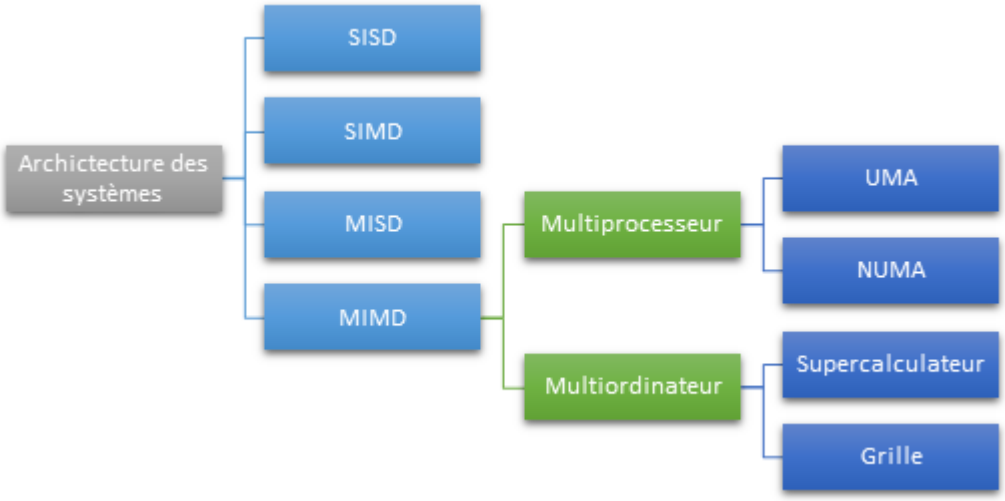


Figure 2.1: Classification des architectures systèmes (Tanenbaum, 2005)

Les systèmes **SISD** (Single Instruction, Single Data) peuvent exécuter un seul flux d'instructions à la fois sur un ensemble unique de données, ce qui correspond aux ordinateurs monoprocesseur qui ne permettent pas de parallélisme. Les systèmes **SIMD** (Single Instruction, Multiple Data) peuvent également exécuter un seul flux d'instructions à la fois, mais sur différents flux de données. Un exemple de système **SIMD** est le réseau de processeurs (Processor Arrays). Les architectures **MISD** (Multiple Instruction, Single Data) sont très rares, mais sont utilisées dans certains domaines sensibles, car ils permettent d'obtenir une redondance des résultats et de réduire le risque d'erreur.

Dans les systèmes **MIMD**, plusieurs processeurs exécutent simultanément différentes séries d'instructions sur différents ensembles de données. Comme le précise Tanenbaum (2005), cette catégorie est composée de deux grandes classes. La première classe comprend les architectures à mémoire partagée que l'on nomme communément les multiprocesseurs qui correspondent à des ensembles de processeurs interconnectés qui partagent un même espace de mémoire. De nos jours, la plupart des ordinateurs personnels possèdent plusieurs processeurs (contenant un ou plusieurs coeurs) et se classent donc dans la famille des systèmes multiprocesseurs. Il existe deux types de systèmes à mémoire partagée : les multiprocesseurs UMA (*Uniform Memory Access*) et les multiprocesseurs NUMA (*Non-Uniform Memory Access*). Les processeurs au sein d'une architecture UMA accèdent à une mémoire continue qui est centralisée et les temps

d'accès mémoire sont sensiblement les mêmes pour chaque processeur. Dans une architecture NUMA, la mémoire est logiquement centralisée, mais physiquement distribuée. Ainsi, les temps d'accès pour un emplacement mémoire diffèrent d'un processeur à l'autre selon la proximité physique. Les multiprocesseurs NUMA permettent ainsi d'exploiter la puissance de centaines de processeurs avec les avantages d'un système à mémoire partagée.

La deuxième classe MIMD est constituée des multiordinateurs. Dans ce type d'architecture, plusieurs processeurs qui possèdent leur propre mémoire communiquent par le biais de messages afin de réaliser un travail commun. Comme le montre la Figure 2.1, les supercalculateurs et les grilles informatiques sont des types de multiordinateur. Les supercalculateurs sont des architectures très répandues de nos jours et possèdent de nombreux processeurs répartis sur différents nœuds de calcul qui communiquent au sein d'un réseau local. Ces nœuds sont généralement des multiprocesseurs composés de plusieurs processeurs qui partagent une mémoire commune. Les processeurs peuvent être de différents types comme des CPUs ou GPUs. De plus, les supercalculateurs sont souvent des systèmes homogènes qui ont l'avantage d'être facilement extensibles et permettent d'obtenir une puissance de calcul très importante.

Durant ces dernières années a eu lieu l'expansion d'Internet et des réseaux WAN (*Wide Area Network*) qui permettent des échanges de données très rapides entre des entités géographiquement dispersées. De ce fait, plusieurs architectures parallèles basées sur ces réseaux ont émergé récemment, par exemple les grilles informatiques (*grid*) et les architectures en nuage (*cloud*). Par ailleurs, les architectures **SPMD** (Single Program, Multiple Data), utilisées principalement par les GPUs, peuvent étendre la classe des architectures MIMD (Brodtkorb *et al.*, 2010). Dans le courant des années 2000, les chercheurs se sont intéressés à la possibilité d'utiliser la puissance des cartes graphiques (GPUs) pour effectuer des calculs généraux. Bien qu'à la base les GPUs soient utilisés principalement pour des calculs graphiques, les architectures récentes des cartes graphiques permettent d'exploiter les nombreux processeurs pour réaliser des traitements divers. Dans une architecture SPMD, les processeurs exécutent le même programme sur des données différentes, mais contrairement aux architectures SIMD, ils n'ont pas besoin d'exécuter la même instruction au même instant. En 2006, NVIDIA crée l'architecture très populaire *CUDA* ainsi que le langage *CUDA C* (NVIDIA, 2007) facilitant la parallélisation de programme sur leurs GPUs. Même si les GPUs ont une architecture SPMD, au sein d'une architecture CUDA, un sous-ensemble de processeurs (warp) s'exécute à la manière d'un système SIMD.

2.1.2 ENVIRONNEMENT LOGICIEL

Les différentes architectures parallèles présentées ont chacune leur environnement logiciel privilégié. L'environnement logiciel peut comprendre un langage de programmation, un compilateur ou un modèle de programmation. Dans un système à mémoire partagée, différents fils d'exécution appelés *threads* peuvent s'exécuter en parallèle au sein d'un processus. Les *threads* ont accès à une mémoire globale commune, mais possèdent leur propre pile d'exécution. Les outils qui permettent de faire de la programmation multithreadée ont différents niveaux d'abstraction qui permettent de cacher certains détails de l'implémentation parallèle. L'interface de programmation **pThreads** (Nichols *et al.*, 1996), adaptée au langage C, a un bas niveau d'abstraction et permet la gestion précise de threads. Le programmeur doit ainsi gérer la création des threads, les zones critiques, les verrous (*mutex*) ou encore les synchronisations entre les threads. À un niveau d'abstraction plus élevé, le langage **JAVA** (JAVA, 2013) permet nativement la manipulation de l'objet Thread et quelques mots clés permettent d'abstraire certains comportements particuliers comme les sections critiques. Enfin, **OpenMP** (OpenMP, 2013) et *Threading Building Blocks* (**TBB**) (TBB, 2006) offrent également un niveau d'abstraction plus élevé que pThreads, ce qui facilite le travail du programmeur mais lui donne un contrôle moins important sur la gestion de son programme. L'utilisation de OpenMP se fait par l'ajout de directives au programme séquentiel original. De ce fait, le type de parallélisation est incrémental car il est possible de transformer le programme séquentiel en un programme parallèle bloc par bloc. Les directives permettent, entre autres, d'indiquer les sections à exécuter en parallèle et d'indiquer la nature des variables (privée ou partagée). La bibliothèque TBB développée en `C++` par INTEL permet une abstraction des détails logiciels et matériels et s'utilise par le biais de fonctions prédéfinies.

Pour les multiordinateurs, les supercalculateurs et les grilles informatiques, les communications se font par échange de messages entre les processeurs. Parmi les langages de programmation permettant de faire du parallélisme à passage de messages, **MPI** (MPI Forum, 2009) est le langage standard. Celui-ci fonctionne avec des langages populaires tels que Fortran, C et C++. Avec MPI, chaque processus va exécuter le même programme, mais différentes opérations pourront être effectuées selon l'identifiant du processus. Ces opérations sont généralement des envois ou réceptions de messages qui peuvent contenir diverses données. De plus, MPI permet la communication d'un processus à un autre (communication point à point) mais aussi la communication entre un ensemble de processus (communication

collective) afin, par exemple, de partager (*scatter*) ou rassembler (*gather*) des données. Pour gérer à la fois la mémoire partagée au sein d'un nœud et la mémoire distribuée entre les nœuds d'un supercalculateur, plusieurs langages tels que MPI, pThreads ou OpenMP peuvent être conjointement utilisés. De plus, les architectures en nuage peuvent être exploitées par le biais de modèles de programmation comme **MapReduce** (Miner et Shook, 2012) et de frameworks comme **Hadoop** (Hadoop, 2015).

Pour les cartes graphiques, le langage et le compilateur utilisés dépendent de l'architecture et du constructeur de la carte graphique. Pour les cartes graphiques NVIDIA ayant une architecture CUDA, NVIDIA a créé le compilateur nommé **CUDA C** (NVIDIA, 2007). La base du langage utilisé est le C auquel seulement quelques mots-clés ont été ajoutés afin de rendre plus simple son utilisation. Par ailleurs, la technologie **ATI Stream** (AMD, 2014) permet de faire des calculs génériques sur des cartes graphiques *AMD*. Enfin, le langage **OpenCL** (Kronos, 2014) permet également la programmation parallèle sur GPU et prend en charge plusieurs types d'architecture.

Le choix du langage de programmation se fait généralement en fonction de l'architecture du système. En effet, si l'architecture utilisée est une architecture à mémoire distribuée, l'utilisation d'un langage à passage de messages tel que *MPI* sera privilégiée. S'il s'agit d'une architecture à mémoire partagée, des langages comme *OpenMP* ou *TBB* pourront convenir. Enfin, s'il s'agit d'un GPU, le langage spécifique associé au modèle du processeur graphique pourra être utilisée ou un langage plus générique tel que *OpenCL*.

2.1.3 MESURE DE L'EFFICACITÉ DU PARALLÉLISME

Dans le but d'analyser les performances d'un programme parallèle, une comparaison avec le temps d'exécution d'une implémentation séquentielle est nécessaire et permet de mettre en évidence les gains apportés par le parallélisme. Basées sur ce concept, les deux principales métriques utilisées pour évaluer les performances d'une parallélisation sont l'accélération et l'efficacité (Rauber et Rünger, 2010). L'accélération (*speedup*) est la proportion entre le temps d'exécution séquentiel T_{seq} et le temps nécessaire à l'algorithme parallèle T_{par}. L'Équation 2.1 définit la mesure d'accélération s.

$$s = \frac{T_{seq}}{T_{par}} \tag{2.1}$$

La seconde métrique est l'**efficacité** (*efficiency*). Contrairement à la mesure d'accélération, elle fait intervenir le nombre de processeurs. L'efficacité e pour p processeurs est définie par l'Équation 2.2.

$$e(p) = \frac{T_{seq}}{p \times T_{par}} \tag{2.2}$$

D'autres mesures peuvent être calculées lors de l'évaluation d'une approche parallèle (Quinn, 2003). Parmi elles, l'**iso-efficacité** (*isoefficiency*) estime l'extensibilité d'un programme. Elle permet notamment de savoir si une efficacité donnée est atteignable. Finalement, en cas de faibles gains de parallélisation, la métrique de **Karp-Flatt** permet d'en déterminer les causes potentielles à savoir un manque d'opportunité de parallélisme ou des surcoûts de communications trop importants.

2.1.4 L'ÉVOLUTION DES SUPERCALCULATEURS ET DU PARALLÉLISME

Dans un monde très concurrentiel, la course à la technologie a mené au financement et à la conception de machines de plus en plus puissantes. Afin de mettre en exergue cette évolution, le projet TOP500 (Strohmaier, 2018) classifie et répertorie les 500 supercalculateurs les plus performants au monde depuis 1993 et est mis à jour deux fois par an. Les machines sont hiérarchisées selon le nombre d'opérations qu'elles sont capables d'effectuer par seconde d'après le test LINPACK (Dongarra, 1979). Les caractéristiques des cinq supercalculateurs les plus puissants du monde selon ce classement sont détaillées au Tableau 2.1. En plus du rang et du nom, l'institut dans lequel se situe la machine, le nombre de nœuds, le nombre de cœurs CPU et la présence ou non de carte graphique dans l'architecture matérielle sont précisés. Par ailleurs, deux métriques qui symbolisent la puissance de calculs sont présentées en Tflop/s (téraflops = mille milliards d'opérations par seconde) : *Rmax* qui correspond à valeur maximale obtenue lors du test LINPACK et *Rpeak* qui se réfère à la performance théorique de la machine.

D'après le classement, le supercalculateur américain Summit est à l'heure actuelle le plus puissant au monde en proposant une puissance de calculs de 122 pétaflops, soit 122 billiards d'opérations par seconde. Comme ses concurrents, il dispose d'un nombre important de cœurs de CPU répartis sur des milliers de nœuds. Avec l'avènement des GPUs pour traiter de calculs génériques au milieu des années 2000, de plus en plus de supercalculateurs sont composés de cartes graphiques pour accroître significativement leur puissance de calculs à moindres

c

Tableau 2.1: Classement des cinq supercalculateurs les plus puissants du monde (Strohmaier, 2018)

Rang	***Nom***	***Institut***	***Nœuds***	***Cœurs de CPU***	***GPUs***	***Rmax (Tflop/s)***	***Rpeak (Tflop/s)***
1	Summit	DOE/SC/Oak Ridge National Laboratory	4608	2282544	oui	122300	187659
2	Sunway TaihuLight	National Supercomputing Center in Wuxi	40960	10649600	non	93014	125435
3	Sierra	DOE/NNSA/LLNL	4474	1572480	oui	71610	119193
4	Tianhe-2A	National Super Computer Center in Guangzhou	17792	4981760	non	61444	100678
5	AI Bridging Cloud Infrastructure	National Institute of Advanced Industrial Science and Technology (AIST)	1088	391680	oui	19880	32576

coûts. Cette tendance est confirmée par le classement présenté puisque trois des cinq machines disposent de GPUs au sein de leurs nœuds.

L'évolution de la capacité de calcul des supercalculateurs au cours des dernières années est significative comme l'expose la Figure 2.2 qui présente les puissances de calculs *Rmax* des machines du TOP500 depuis 1993, en particulier celles du rang 1 et du rang 500 ainsi que la puissance cumulée des 500 meilleurs supercalculateurs. Le principal constat qui peut être effectué est que la puissance de la meilleure machine est en constante augmentation depuis la création du classement. En effet, si le premier rang atteignait 100 GFlop/s (100 milliards d'opérations par seconde) en 1993, la barre du TFlop/s a été dépassée quelques années plus tard. En outre, la premier supercalculateur qui a disposé d'une puissance de calculs de plus d'un pétaflops a été construit en 2008. D'après la projection de la courbe d'évolution, l'exaflops devrait être atteint au début des années 2020 comme le confirme la course aux machines exaflopiques à laquelle la société Bull participe avec son supercalculateur BullSequana (Bull, 2018).

Alors que les machines parallèles sont de plus en plus puissantes et accessibles de nos jours, il n'est pas aisé d'exploiter au maximum leur puissance. Néanmoins, des avancées

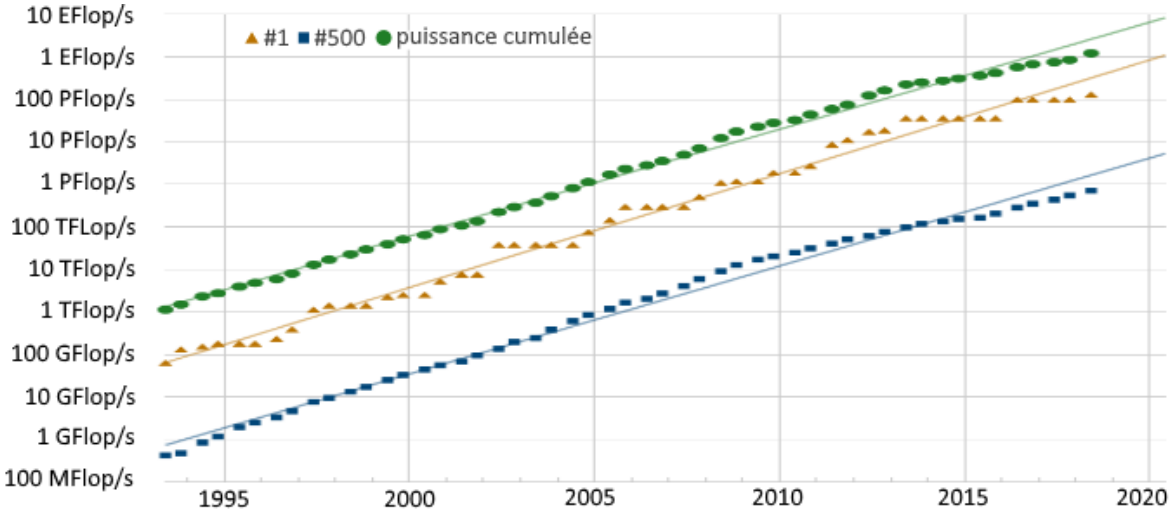


Figure 2.2: Évolution de la puissance de calculs des supercalculateurs du TOP500 (Strohmaier, 2018)

considérables ont été réalisées pour concevoir des programmes parallèles performants et profiter des nouvelles architectures hybrides. Dans le domaine de la chimie, une des plus grandes simulations a été effectuée par Zhao *et al.* (2013) et avait pour but de reproduire le comportement de 64 millions d'atomes qui constituent une des parties du virus du SIDA. Deux années de calculs sur le supercalculateur Blue Waters, qui est doté d'une puissance de calculs de 11,6 pétaflops, ont été nécessaires pour simuler $1,2$ microsecondes. Ces travaux, qui permettent de mieux comprendre le fonctionnement des molécules du SIDA, n'auraient pas pu être envisagés il y a 20 ans en raison du manque de puissance de calculs ou même du manque d'espace mémoire des machines de l'époque. Par ailleurs, dans le domaine de la recherche opérationnelle, le problème classique de voyageur de commerce est un des problèmes les plus étudiés. Si dans les années 90, les instances résolues dans un temps raisonnable étaient constituées de seulement plusieurs dizaines de villes, il est de nos jours possible de traiter des centaines voire des milliers de villes en quelques secondes ou quelques minutes, en s'appuyant notamment sur la puissance des GPUs (Delévacq *et al.*, 2013).

De façon générale, les architectures parallèles permettent de répondre à des problématiques de plus en plus complexes dans des domaines variés. La section suivante décrit la manière dont sont exploitées ces architectures pour améliorer et accélérer les métaheuristiques multi-objectifs.

2.2 LES MÉTAHEURISTIQUES PARALLÈLES MULTI-OBJECTIFS

Bien qu'il existe des lignes directrices pour paralléliser un algorithme quelconque (Foster, 1995), les métaheuristiques et plus particulièrement les métaheuristiques multi-objectifs ont fait l'objet de nombreuses approches de parallélisation. Selon la classification réalisée par Veldhuizen *et al.* (2003), il existe quatre grands paradigmes pour les AEMO parallèles : le modèle maître-esclave, le modèle en îles, le modèle par diffusion et le modèle hybride. Ces paradigmes se divisent en plusieurs catégories qui sont exposées à la Figure 2.3.

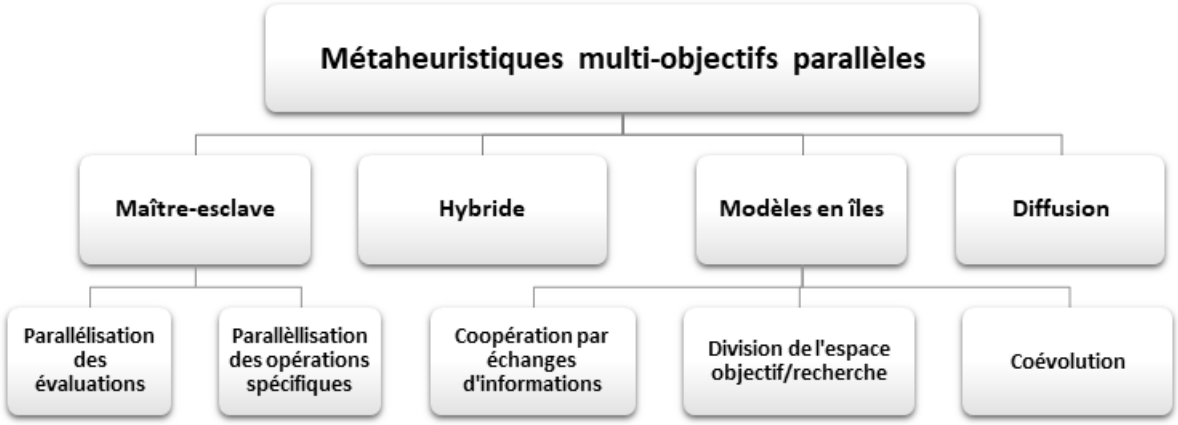


Figure 2.3: Classification des modèles parallèles pour les métaheuristiques multi-objectifs (Veldhuizen *et al.*, 2003)

Parmi les modèles parallèles de la littérature, certains peuvent s'appliquer autant aux algorithmes uni-objectif qu'aux algorithmes multi-objectifs, alors que d'autres ont été conçus en fonction des spécificités de l'optimisation multi-objectifs, comme la gestion d'une archive de solutions non-dominées. Les métaheuristiques basées sur une population sont particulièrement adaptées pour le parallélisme, car les opérations principales (comme le croisement ou les évaluations des fonctions objectif) peuvent être traitées de manière indépendante pour chaque individu. C'est à partir de ce constat que des modèles maître-esclave ont été proposés. De plus, le parallélisme n'est pas seulement utilisé pour résoudre un problème plus rapidement, mais il peut permettre de développer des modèles plus efficaces (Luna et Alba, 2015). Ainsi,

un modèle d'algorithme évolutionnaire parallèle peut être plus performant qu'un modèle séquentiel même s'ils effectuent le même nombre d'opérations. Par ailleurs, comme les approches multi-objectifs auxquelles nous nous intéressons cherchent à obtenir un ensemble de solutions Pareto optimales, certains modèles en îles cherchent à faire en sorte que chaque processeur s'occupe d'un sous-ensemble du front Pareto. Comme le rapportent Luna et Alba (2015), la majeure partie des travaux de la littérature concerne le paradigme maître-esclave et le paradigme en îles. En effet, les pourcentages d'utilisation des différents modèles exposés à la Figure 2.4, montrent que près de la moitié des modèles proposés sont des modèles en îles et que 45% sont des modèles maître-esclave.

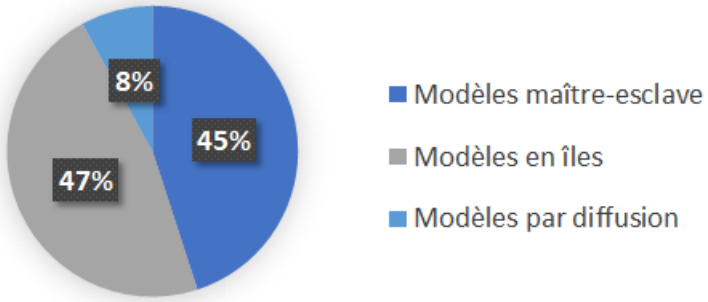


Figure 2.4: Pourcentage d'utilisation des modèles parallèles pour les AEMO dans la littérature (Luna et Alba, 2015)

Très récemment, une nouvelle classification a été proposée par Talbi (2018). Dans celle-ci, les modèles parallèles multi-objectifs sont différenciés selon le niveau auquel le parallélisme intervient. Les différents niveaux sont : la solution, l'itération et l'algorithme (l'AEMO). Les deux premiers niveaux s'intègrent dans le paradigme maître-esclave alors que le niveau algorithmique correspond aux modèles en îles et aux modèles par diffusion. Les caractéristiques des trois classes sont résumées dans le Tableau 2.2. La granularité, qui estime la taille des tâches parallèles, et l'extensibilité permettent d'associer les classes de modèle à des paradigmes de parallélisation et des architectures spécifiques. Par ailleurs, les trois classes ont un objectif d'accélération et seule la parallélisation au niveau de l'algorithme vise à améliorer la qualité des solutions obtenues en modifiant le comportement de l'AEMO.

Les prochaines sections présenteront les différents paradigmes, tels que présentés à la Figure 2.3 et les principaux travaux dont ils font l'objet. Les modèles présentés sont les modèles qui ne sont pas spécifiques à la résolution de problèmes continus et qui permettent de résoudre des problèmes combinatoires.

Tableau 2.2: Caractéristiques des classes de modèles parallèles pour les AEMO (Talbi, 2018)

Caractéristiques	Niveau Solution	Niveau Itération	Niveau Algorithme
Granularité	Grain fin (dépendances entre les sous-fonctions)	Grain moyen (nombre d'individus)	Gros grain (taille et fréquence des échanges)
Extensibilité	Nombre de sous-fonctions	Taille de la population	Nombre de AEMO
Comportement de l'AEMO	Inchangé	Inchangé	Modifié
Objectifs	Accélération	Accélération	Qualité et accélération

2.3 LE PARADIGME MAÎTRE-ESCLAVE

Dans le modèle maître-esclave, un processus maître exécute une métaheuristique séquentielle et distribue le travail de certaines opérations sur plusieurs processus esclaves. Ces derniers effectuent alors les différents calculs et retournent les résultats au processus maître qui peut alors poursuivre son déroulement. Le comportement de l'algorithme n'est pas modifié et seule la diminution du temps d'exécution est visée. De ce fait, la qualité des solutions reste inchangée. Dans ce type d'approche, l'AEMO est généralement parallélisé au niveau de l'itération, selon la classification de Talbi (2018), puisque la population est évaluée de manière distribuée. L'extensibilité du modèle est alors limitée par le nombre d'individus de la population. Si l'évaluation d'une solution prend un temps excessivement long, l'AEMO peut être parallélisé au niveau de la solution. La granularité de l'algorithme et l'extensibilité du modèle dépendent alors étroitement du temps d'exécution des sous-fonctions nécessaires au calcul des fonctions objectif.

2.3.1 PARALLÉLISATION DES ÉVALUATIONS

Pour les métaheuristiques multi-objectifs, ce sont les évaluations des solutions qui sont généralement les plus coûteuses en temps (Veldhuizen *et al.*, 2003). Les temps d'exécution des opérations spécifiques telles que les croisements ou les mutations sont souvent ignorés. Naturellement, les évaluations des fonctions objectif doivent être assez complexes et longues afin obtenir des accélérations significatives.

Il existe trois méthodes pour paralléliser la phase d'évaluation d'une population (Veldhuizen *et al.*, 2003). Dans la première méthode, illustrée à la Figure 2.5a, la population est subdivisée et chaque processeur (de 1 à k) calcule l'ensemble des fonctions objectif des individus qui lui sont assignés. La seconde méthode associe chaque processeur P à une fonction objectif f unique (P_1 évalue f_1, ..., P_M évalue f_M, avec M le nombre d'objectifs). Ainsi, un processeur calcule une des fonctions objectif pour l'ensemble des solutions ou une partie d'entre elles. Cette méthode, illustrée à la Figure 2.5b, est seulement réalisable quand les fonctions objectif sont indépendantes les unes des autres. Enfin, la dernière méthode consiste à décomposer l'évaluation des fonctions objectif sur plusieurs processeurs. Chaque processeur calcule une partie d'une fonction objectif qui sera recombinée par le processus maître. Cette décomposition peut se faire par un découpage des données ou par un découpage fonctionnel.

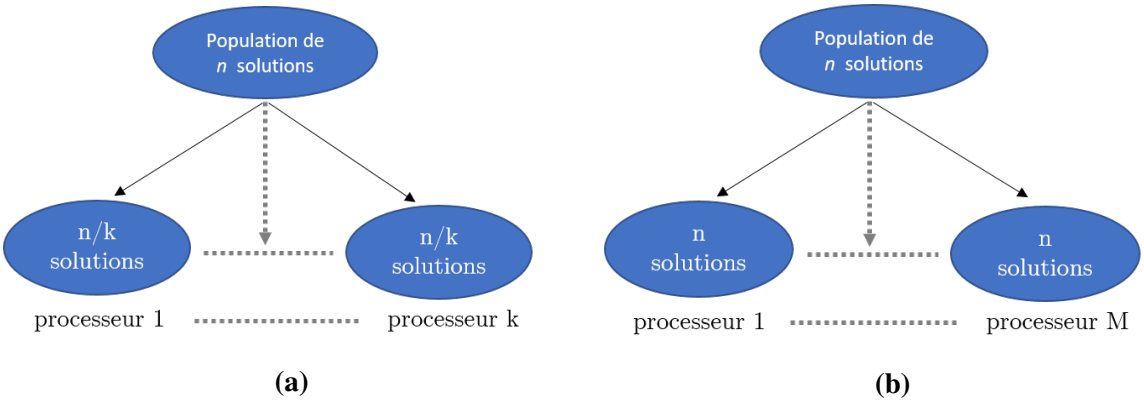


Figure 2.5: Schéma de fonctionnement de deux méthodes de parallélisation d'évaluation

Il y a quelques années, Durillo *et al.* (2008) ont étudié plusieurs méthodes pour paralléliser l'algorithme de référence NSGA-II en suivant le modèle maître-esclave. L'utilisation d'une version stationnaire de l'algorithme et l'implémentation de communications asynchrones permettent d'obtenir des accélérations importantes. En effet, sur un problème multi-objectifs gourmand en matière d'évaluation, une accélération de 106 est observée avec 286 processeurs. Ferringer *et al.* (2009) ont également proposé un modèle parallèle asynchrone LC-ε-NSGA-2 adapté à un algorithme multi-objectifs, en l'occurrence ε-NSGA-2, une version modifiée de NSGA-II. Dans LC-ε-NSGA-2, la population Enfant est composée de deux listes : la liste des individus restants à évaluer et la liste des individus déjà évalués. Lors de la phase de remplacement gérée par le processus maître, les individus de la population Parent et de la liste des individus Enfants déjà évalués sont considérés.

Mostaghim *et al.* (2008) décrivent une méthode our paralléliser efficacement un algorithme par essaim particulaire multi-objectifs avec des ressources hétérogènes. Le processus maître répartit les évaluations des solutions sur les différents processeurs. Dès qu'il reçoit un nombre N_s de solutions évaluées, il continue son travail et attend les prochaines évaluations. La phase d'expérimentation a révélé des gains de parallélisation importants. Cependant, cet algorithme nécessite de calibrer au mieux le nombre N_s afin d'obtenir les meilleures performances possibles. Plus récemment, Depolli *et al.* (2013) proposent une méthode pour paralléliser efficacement un algorithme d'évolution différentielle. Des communications asynchrones sont utilisées pour évaluer les solutions dans une version stationnaire de l'algorithme. Une accélération presque linéaire est obtenue avec l'utilisation de quatre cœurs de processeur.

2.3.2 PARALLÉLISATION DES OPÉRATIONS SPÉCIFIQUES

Même si dans le paradigme maître-esclave, la majeure partie des travaux se concentrent sur la parallélisation des évaluations, il est possible d'accélérer d'autres opérations gourmandes en temps d'exécution. Les opérations spécifiques à la métaheuristique utilisée peuvent être parallélisées telles que les croisements pour les algorithmes évolutionnaires ou le calcul d'un nouveau voisin pour les algorithmes de recherche locale.

Dans le cadre des AEMO, une des parties les plus coûteuses en temps est l'étape d'assignation du facteur de dominance (Talbi, 2009). Différents auteurs se sont intéressés à la parallélisation de la méthode d'assignation de rang. Tout d'abord, Lančinskas et Žilinskas (2012) affirment que, lorsqu'il faut évaluer deux populations conjointes (Parent et Enfant), les comparaisons nécessaires à l'assignation de rang peuvent être fragmentées. Ainsi, ils proposent une approche dans laquelle chaque processeur à besoin de connaitre uniquement la population Parent. Pour distribuer la population Parent sur tous les processeurs et pour récupérer l'ensemble des résultats sur le processus maître, une topologie en hypercube peut être utilisée afin de réduire les temps de communication. D'après la phase d'expérimentation, cette méthode de parallélisation permet d'obtenir une accélération de 23 avec 64 processeurs pour un problème continu dont la phase d'évaluation est conséquente. Gupta et Tan (2015) ont proposé une implémentation parallèle sur GPU de l'algorithme NSGA-II en se focalisant sur l'assignation de rang. Dans celle-ci, chaque thread est en charge de calculer le rang d'un individu particulier. Moreno *et al.* (2017) ont également exploité la puissance de calcul des GPUs pour paralléliser une version améliorée de l'assignation de rang.

Par ailleurs, une des premières parallélisations d'un algorithme par essaim particulaire multi-objectifs sur GPU a été proposée par Zhou et Tan (2009). Dans celle-ci, ils utilisent une version modifiée de l'algorithme VEPSO (Parsopoulos et Vrahatis, 2002). La plupart des opérations (calcul de fitness, déplacement d'une particule, etc.) se font sur le GPU. Les résultats obtenus montrent une augmentation linéaire de l'accélération avec le nombre de particules utilisées. Cependant, l'utilisation d'un grand nombre de particules n'améliore plus la qualité des solutions.

2.4 LE PARADIGME EN ÎLES

Le paradigme en îles, illustré à la Figure 2.6 , est l'un des paradigmes de parallélisme les plus étudiés pour les AEMO (León *et al.*, 2009). Dans un tel modèle, plusieurs sous-populations, une par processeur, évoluent de manière relativement autonome. Chacun des processeurs exécute un algorithme évolutionnaire de manière classique. Des algorithmes, des opérateurs et des paramètres différents peuvent être utilisés pour chaque population. Périodiquement, des échanges d'informations interviennent entre les processeurs. Dans un contexte uni-objectif, ce sont généralement les bonnes solutions (meilleures fitness) qui sont échangées entre les populations, on parle alors de migrations d'individus. Les individus échangés sont ensuite intégrés dans de nouvelles populations, ce qui permet d'améliorer l'exploration des algorithmes en introduisant des individus externes, mais également l'intensification puisqu'il s'agit des meilleurs individus. Dans un contexte multi-objectifs, les solutions échangées peuvent être une partie des solutions non-dominées. De plus, comme le but est d'obtenir de multiples solutions, c'est-à-dire le front Pareto, il convient de faire en sorte que chaque processeur s'occupe de trouver une partie spécifique de ce front.

Comme l'expose la Figure 2.3, il existe trois catégories de modèles en îles pour les AEMO. Dans la première catégorie, les îles s'échangent des informations et essaient de coopérer afin d'améliorer le processus de recherche. Dans la seconde catégorie, un processus maître découpe l'espace de recherche ou l'espace objectif et attribue une région spécifique à chaque processeur. Enfin, la dernière catégorie concerne les méthodes de coévolution coopérative qui cherchent à diviser le problème initial en sous-problèmes. Dans la suite de cette section, ces catégories de modèles conçues pour les algorithmes évolutionnaires sont présentées. La dernière partie de cette section se consacre aux modèles en îles adaptés aux autres métaheuristiques.

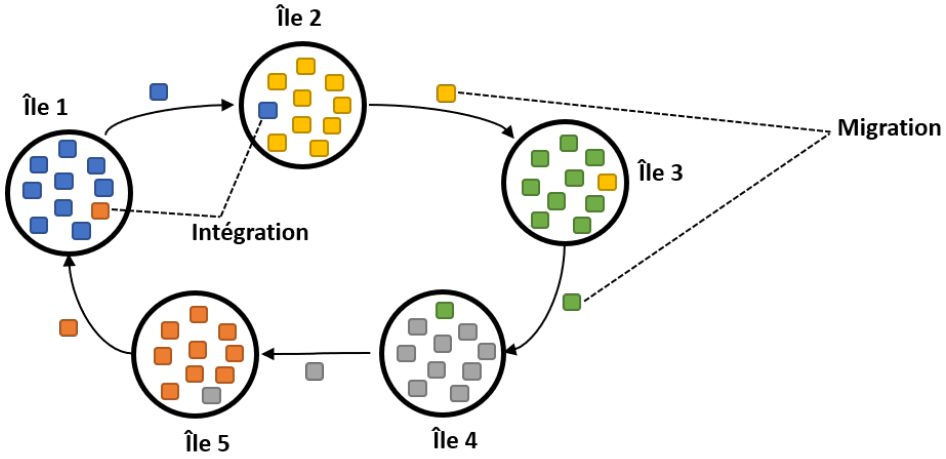


Figure 2.6: Exemple de modèle à 5 îles avec une topologie en anneau

2.4.1 COOPÉRATION PAR ÉCHANGES D'INFORMATIONS

L'un des premiers algorithmes génétiques multi-objectifs utilisant un modèle en îles est l'algorithme DRMOGA proposé par Hiroyasu *et al.* (2000). Le but de ce dernier est d'attribuer à chaque processeur un sous-ensemble de la population afin d'orienter sa recherche vers une partie spécifique du front Pareto. À toutes les k générations, une population globale est reconstituée à partir des sous-populations obtenues par les p processeurs. La population globale est triée selon une unique fonction objectif f_i et p sous-groupes de solutions sont formés selon leur valeur de f_i. Chaque processeur est alors en charge d'un sous-groupe de solutions. La fonction objectif f_i qui sert de référence pour le tri change à toutes les k générations. Cette méthode ne garantit pas l'exploration efficace du front Pareto, mais tente tout de même de spécialiser dynamiquement les processeurs vers des zones spécifiques de l'espace objectif. La Figure 2.7 présente un exemple de découpage de l'espace objectif par l'algorithme DRMOGA. La fonction objectif qui sert de référence est ici f_1. De Toro Negro *et al.* (2004) utilisent une approche similaire, nommée PSFGA, pour attribuer une partie de l'espace de recherche à chaque processeur. L'efficacité de la parallélisation est exposée avec l'utilisation de 4 processeurs.

Dans l'algorithme MFED proposé par Essabri *et al.* (2006), ce sont les premiers fronts obtenus par une assignation de rang qui sont échangés entre les îles. Un processeur, nommé organisateur, est en charge de regrouper les fronts et de les redistribuer. Le principal inconvénient de cette méthode est qu'il est difficile de définir le nombre de fronts à échanger, car il

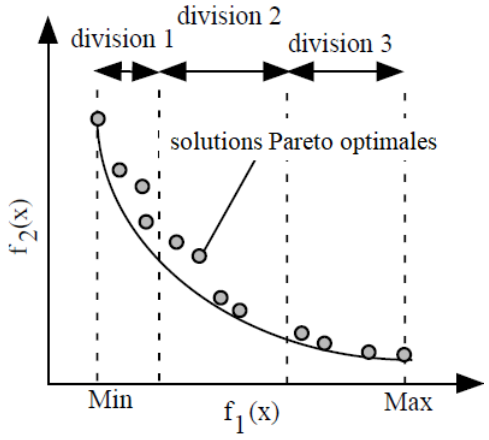


Figure 2.7: Exemple de division d'un espace objectif pour un problème de minimisation bi-objectif par l'algorithme DRMOGA (Hiroyasu *et al.*, 2000)

dépend de la forme du front Pareto. Fernández *et al.* (2013) proposent un modèle parallèle pour un algorithme mémétique dans lequel les solutions non-dominées trouvées par les îles sont échangées selon une topologie en anneau. Les solutions reçues sont alors intégrées à la population d'une île selon un facteur de dominance. Lors de la phase d'expérimentation, quatre îles sont utilisées. La version parallèle du modèle obtient une meilleure qualité de solution que la version séquentielle à nombre égal d'évaluations. Cependant, aucune information sur les temps d'exécution n'est précisée.

À la manière de certains algorithmes many-objectives présentés dans le premier chapitre, l'algorithme PMRPEA proposé par Figueira *et al.* (2010), utilise des points de référence dans sa résolution. Après avoir calculé ces points dans une première étape, ceux-ci sont associés à une fonction d'agrégation avec différents vecteurs de poids. La phase d'expérimentation révèle qu'à temps d'exécution égal, l'algorithme PMRPEA obtient, avec dix points de référence, des solutions significativement meilleures que des algorithmes comme NSGA-II ou IBEA. Comme le soulignent les auteurs, des analyses sur l'accélération obtenue par la méthode restent à effectuer.

Pilat et Neruda (2010) proposent un modèle en îles dans lequel certaines îles exécutent une métaheuristique multi-objectifs alors que d'autres traitent le problème de manière uni-objectif. À intervalles réguliers, les îles multi-objectifs envoient leurs meilleures solutions aux îles uni-objectif. Les résultats de l'expérimentation montrent que l'utilisation d'une seule île uni-objectif avec neuf îles multi-objectifs permet d'obtenir les meilleures performances. Aucune comparaison à des modèles de la littérature et aucune mention d'accélération n'est

faite dans l'article.

L'algorithme PasMoQAP, proposé par Sanhueza *et al.* (2017), utilise un modèle en îles avec un algorithme mémétique pour résoudre le problème d'affectation quadratique multi-objectifs. L'expérimentation permet de mettre en évidence la supériorité de leur approche comparée à un modèle en îles contenant NSGA-II. Enfin, Hernández Gómez *et al.* (2016) proposent une version parallèle de l'algorithme SMS-MOEA (présenté dans la Section 1.4.2). Dans leur modèle, des populations de faible taille sont utilisées et communiquent de manière asynchrone. L'étude expérimentale révèle que l'approche parallèle a une convergence plus rapide et une meilleure robustesse des résultats que la version séquentielle dans la résolution de problèmes many-objectives.

2.4.2 DIVISION DE L'ESPACE DE RECHERCHE/OBJECTIF EXPLICITE

Certains modèles de parallélisation proposent de découper explicitement l'espace de recherche ou l'espace objectif en plusieurs parties. Ainsi, chaque processeur est en charge de trouver des solutions contenues dans la partie qui lui a été attribuée. Cela permet de distribuer efficacement le travail sur les processeurs, mais les gains d'accélération peuvent dépendre des caractéristiques du front Pareto, c'est-à-dire la convexité et la continuité (Mishra *et al.*, 2011).

Le modèle de séparation en cônes, proposé par Branke *et al.* (2004), est le premier à attribuer une partie de l'espace objectif à chaque processeur. À intervalles réguliers, le point Nadir est utilisé comme point de référence pour construire les cônes et définir les zones spécifiques. Pour ce faire, l'angle droit formé par le point Nadir et les solutions extrêmes est subdivisé en plusieurs parties égales. Les solutions qui ne se trouvent pas dans la région de l'espace objectif attribué au processeur sont considérées comme infaisables. Dans ce cas, afin de conserver les bonnes solutions, elles seront tout de même transférées au processeur qui s'occupe de l'espace en question. Ce modèle permet à chaque processeur d'explorer potentiellement n'importe quelle partie de l'espace de recherche. La Figure 2.8 illustre un exemple d'une séparation en cônes pour trois processeurs dans le cadre d'un problème bi-objectif. Chaque processeur est responsable d'une région spécifique $P1$, $P2$ ou $P3$. La phase d'expérimentation a révélé que ce modèle obtient de très bonnes performances pour un problème bi-objectif. Cependant, l'adaptation à un problème à trois objectifs est plus difficile puisque les résultats obtenus montrent une mauvaise distribution des solutions.

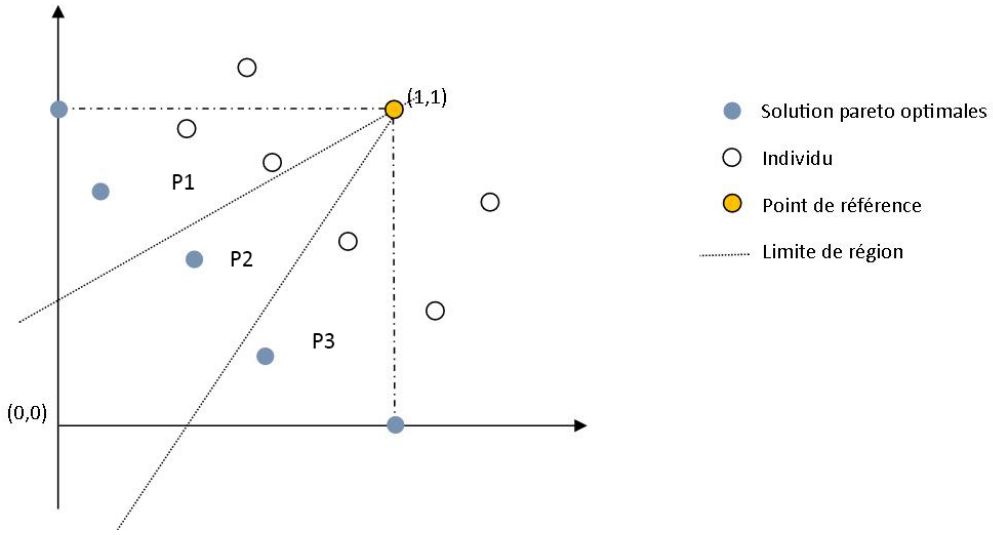
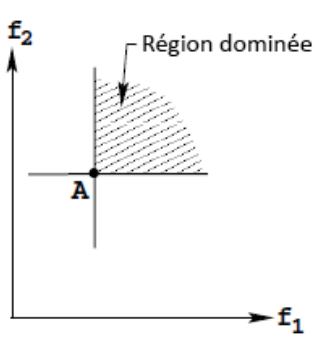


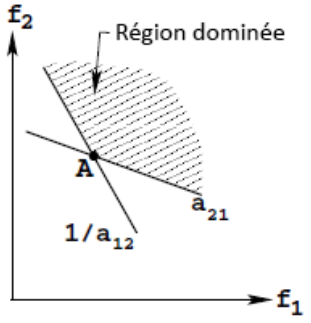
Figure 2.8: Séparation de l'espace objectif en cônes pour un problème de minimisation bi-objectif

Au milieu des années 2000, Streichert *et al.* (2005) proposent d'utiliser les méthodes de regroupement (*clustering*) pour paralléliser les algorithmes évolutionnaires multi-objectifs. Deux approches sont proposées : le découpage de l'espace de recherche et le découpage de l'espace objectif. Dans le modèle proposé, les solutions Pareto optimales sont rassemblées sur une île, partitionnées grâce à l'algorithme en k-moyennes puis redistribuées sur les îles disponibles. Tout comme la méthode de séparation en cônes, une solution qui n'appartient pas à la zone traitée par le processeur est déclarée infaisable. Selon l'expérimentation effectuée, cette méthode est plus efficace que la méthode de séparation en cônes et présente l'avantage de s'étendre facilement à plus de deux objectifs.

Deb *et al.* (2003) proposent une autre approche pour répartir le calcul du front Pareto entre les différents processeurs. Dans celle-ci, la notion de dominance guidée, décrite par Branke *et al.* (2001), est utilisée pour modifier la zone de dominance d'une solution. La Figure 2.9 présente des exemples d'espaces de dominance d'une solution selon les principes de Pareto (2.9a) et selon une fonction pondérée quelconque (2.9b). Cette dominance guidée permet ainsi de définir des régions spécifiques pour chaque processeur en choisissant des pondérations appropriées. Pour définir ces pondérations, seules quelques solutions non-dominées sont échangées entre les processeurs. La méthode a été testée avec l'algorithme NSGA-II et a montré sa faculté à converger vers le front Pareto tout en diminuant les temps d'exécution de manière significative en utilisant jusqu'à six processeurs.



(a) Dominance au sens Pareto



(b) Dominance selon une fonction pondérée quelconque

Figure 2.9: Exemple de différents espaces de dominance d'une solution (Deb *et al.*, 2003)

Bui *et al.* (2009) proposent d'attribuer une partie de l'espace de recherche à chaque processeur. Un processeur cherche des solutions appartenant à un sous-espace particulier nommé hypersphère. Après plusieurs itérations, les hypersphères se déplacent à la manière des particules dans les algorithmes d'essaim particulaire. Pour leur déplacement, des informations locales et globales sont utilisées afin de modifier la trajectoire de l'hypersphère et de l'orienter vers les meilleures solutions. Les informations globales sont échangées au travers d'une archive globale. L'algorithme proposé obtient une meilleure convergence vers le front Pareto que d'autres algorithmes tels que NSGA-II ou des modèles en îles pour un même nombre d'évaluations. Cependant, les temps d'exécution obtenus et les gains de parallélisation ne sont pas indiqués.

2.4.3 COÉVOLUTION COOPÉRATIVE

Dans les algorithmes de coévolution coopérative, le problème à résoudre est divisé en sous-composantes et chacune d'entre elles est résolue de manière indépendante par une population qui est gérée par un processeur. Dans un contexte uni-objectif, Potter et Jong (1994) ont été les premiers à proposer un tel modèle. Les variables de décision sont découpées en sous-ensembles et chaque population cherche à optimiser son ensemble en appliquant un algorithme évolutionnaire standard. Pour évaluer la qualité de son optimisation, une population construit une solution complète à partir des meilleurs sous-ensembles des autres populations et de son sous-ensemble de décision dédié. Pour une population donnée, les solutions sont composées d'une part fixe et d'une part variable à optimiser. Pour adapter le modèle de coévolution

coopérative à l'optimisation multi-objectifs, Dorronsoro *et al.* (2013) proposent un nouveau modèle CCMOEA. La principale différence avec l'optimisation uni-objectif est qu'il n'y a pas de meilleure solution unique pour chaque sous-population. Plusieurs solutions non-dominées sont ainsi échangées entre les populations. Un exemple de fonctionnement du modèle avec trois populations est présenté à la Figure 2.10. Dans cet exemple, le nombre de solutions partielles à échanger est fixé à quatre. Les populations partagent leur meilleure solution partielle bdv_i avec les autres populations et optimisent leur propre partie variable dv_i. À chaque évaluation d'une solution, la population choisit au hasard parmi les solutions partielles qu'elle a reçues. Si le nombre de solutions non-dominées à envoyer n'est pas suffisant, des solutions dominées appartenant à la population sont choisies aléatoirement. D'après les résultats obtenus lors de la phase d'expérimentation, la méthode proposée permet d'obtenir une accélération linéaire pour de petites et grosses instances d'un problème d'ordonnancement nommé RSMP (Robust Static Mapping Problem).

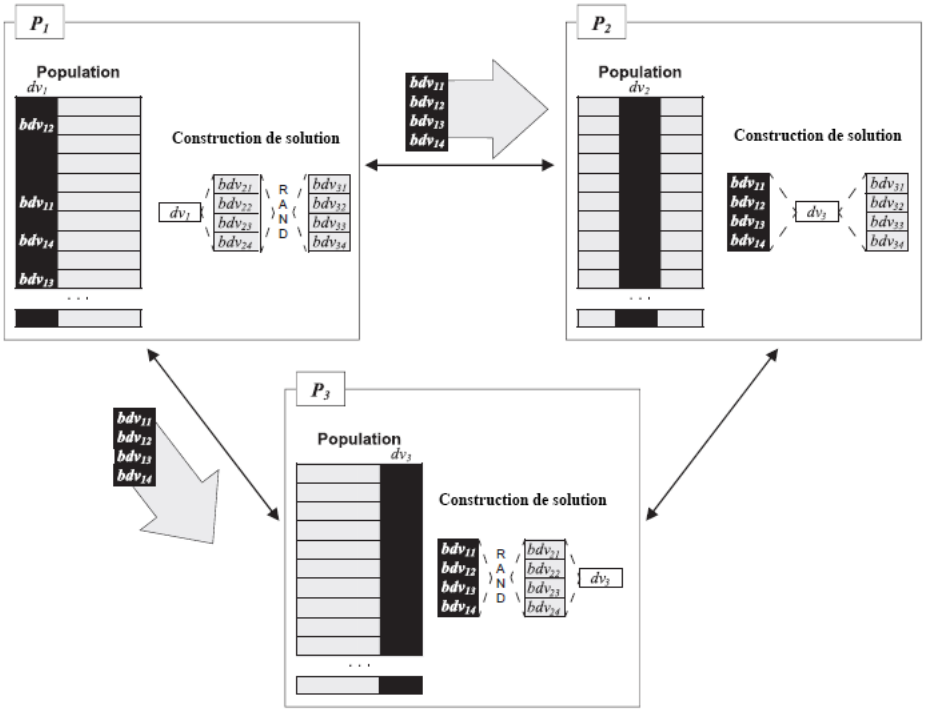


Figure 2.10: Schéma de fonctionnement d'un algorithme de coévolution coopérative multi-objectifs à 3 populations (Dorronsoro *et al.*, 2013)

2.4.4 ADAPTATION DU MODÈLE EN ÎLES AUX AUTRES MÉTAHEURISTIQUES

Certains travaux ont été effectués afin d'adapter le modèle en îles à des métaheuristiques autres que les algorithmes génétiques. Les travaux parmi les plus significatifs concernant les algorithmes de colonies de fourmis, les algorithmes par essaim particulaire et les algorithmes du recuit simulé sont présentés dans la suite.

Pour paralléliser un algorithme de colonies de fourmis multi-objectifs, Mora *et al.* (2013) proposent un nouveau modèle en îles. Ce dernier attribue à chaque processeur un ensemble de fourmis, nommé colonie. Le modèle cherche à faire en sorte que chaque processeur découvre une partie différente du front Pareto. Pour ce faire, chaque colonie accorde une importance différente à chaque fonction objectif par le biais de la méthode de la somme pondérée. Des migrations de fourmis interviennent entre les colonies, mais seulement dans une direction comme le présente la Figure 2.11. La fourmi qui migre est celle qui est associée à la meilleure solution en tenant compte des pondérations utilisées par la colonie. Les expérimentations ont révélé que ce modèle permet des gains significatifs en termes de qualité de solution et de temps d'exécution.

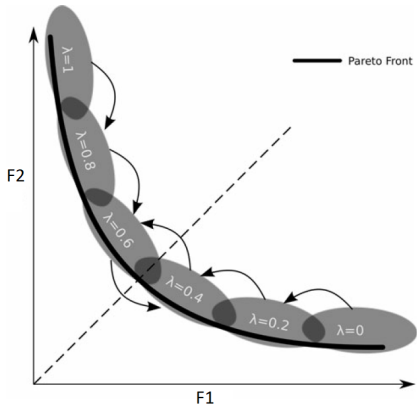


Figure 2.11: Exemple de répartition des espaces pour les colonies de fourmis (Mora *et al.*, 2013)

Mostaghim *et al.* (2007) proposent d'adapter le modèle en îles aux algorithmes par essaim particulaire en attribuant un sous-essaim à chacun des processeurs disponibles. Deux méthodes sont expérimentées pour répartir la recherche du front Pareto. Dans la première approche, nommée clustering C-MOPSO, p groupes de solutions sont formés à partir de l'archive et sont envoyés aux p processeurs disponibles. Dans la seconde approche, nommée hypervolume H-MOPSO, une particule guide est sélectionnée pour chaque sous-essaim selon sa métrique

d'hypervolume marginal. Cette dernière correspond à la différence entre le volume de l'espace dominé par la particule et le volume de l'espace dominé par les autres particules. Les particules ayant le meilleur hypervolume marginal sont sélectionnées comme guides pour les sous-essaims. À la différence de l'approche C-MOPSO, le H-MOPSO peut s'exécuter de manière asynchrone. À chaque fois qu'un sous-essaim renvoie une solution, une nouvelle solution guide est calculée pour lui être renvoyée. Le H-MOPSO peut ainsi très bien évoluer dans un milieu hétérogène. D'après l'expérimentation effectuée, l'approche H-MOPSO est meilleure que l'approche C-MOPSO que ce soit dans un système hétérogène ou un système homogène. L'algorithme PDL-MOEA (Yu *et al.*, 2017) combine à la fois un algorithme par essaim particulaire avec des techniques de décomposition de problème multi-objectifs. Il permet de résoudre de façon parallèle un problème robuste qui a été transformé en problème bi-objectifs. L'expérimentation a permis de constater l'aptitude de PDL-MOEA à trouver de meilleures solutions que d'autres algorithmes reconnus performants en utilisant 5 processeurs.

Banos *et al.* (2006) décrivent deux modèles en îles pour un algorithme de recuit simulé multi-objectifs. Leurs approches utilisent un recuit simulé basé sur une population appelée PSA (*Pareto Simulated Annealing*) (Czyżżak et Jaszkiewicz, 1998). Le premier modèle est un modèle en îles classique dans lequel chaque processeur exécute un algorithme PSA. De temps à autre, des migrations interviennent. L'autre modèle utilise les techniques de découpage de l'espace objectif pour attribuer à chaque population seulement une région à explorer. Les deux modèles ont été comparés lors de la phase d'expérimentation. Cette dernière atteste que le modèle qui divise l'espace objectif obtient des solutions plus proches du front Pareto. De plus, les deux modèles permettent d'obtenir des accélérations significatives en utilisant entre deux et seize processeurs.

2.5 LES AUTRES PARADIGMES

Dans cette section, les modèles de parallélisme hybrides et le modèle par diffusion sont brièvement présentés. Afin de paralléliser les algorithmes multi-objectifs, il est possible de combiner les différentes approches décrites précédemment. Ainsi, Cantu-Paz (2000) définit trois modèles hybrides basés sur un modèle en îles. Dans chacun de ces modèles, chaque nœud de calcul représente une île et chaque île contient soit un modèle maître-esclave, soit un modèle par diffusion ou soit un autre modèle en îles. L'algorithme HybJacIsCone proposé par Gourisaria *et al.* (2013) hybride deux modèles : le modèle en îles et le modèle de séparation

en cônes. Au plus haut niveau de l'algorithme, un modèle en îles s'exécute. Sur chaque île un modèle de séparation en cônes, organisé par un processus maître, est appliqué. À intervalles réguliers, les meilleures solutions sont échangées selon une topologie en anneau comme le présente la Figure 2.12. Le processus maître est alors en charge de répartir les meilleures solutions reçues sur les différents processus esclaves. Lors de l'expérimentation, le modèle a été testé sur un problème de sac à dos multi-objectifs. Les résultats montrent une amélioration significative de la convergence vers le front Pareto comparativement à d'autres modèles de parallélisation existants comme l'approche de séparation en cônes ou le modèle en île classique.

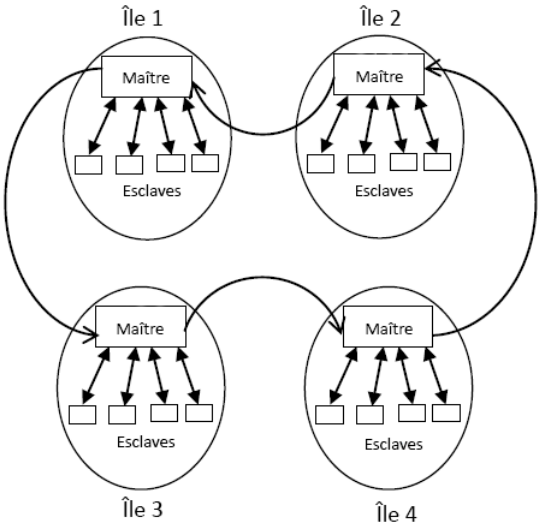


Figure 2.12: Modèle de parallélisation HybJacIsCone (Gourisaria *et al.*, 2013)

L'approche proposée par Jagtap *et al.* (2011) utilise également un modèle en îles. Cette fois-ci, chaque île accélère son processus de recherche par le biais d'un modèle maître-esclave nommé modèle de Jakobovic (Golub et Jakobovic, 2000). Dans ce dernier, les processus esclaves font évoluer entièrement une sous-population, c'est-à-dire qu'ils appliquent l'ensemble des opérations génétiques d'une métaheuristique. Le processus maître est en charge de collecter les meilleurs individus parmi ses esclaves. Toutes les dix générations, les meilleurs individus de chaque île sont échangés entre les îles et partagés par le processus maître sur les processus esclaves. Les résultats obtenus sur un problème de sac à dos multi-objectifs montrent une meilleure convergence et une meilleure diversité des solutions obtenues par rapport à un modèle maître-esclave. De plus, augmenter le nombre de processeurs de deux à

quatre permet également d'améliorer la qualité des solutions obtenues.

À la différence du modèle en îles, le modèle par diffusion, également appelé modèle cellulaire, représente un parallélisme à grain fin puisqu'il ne fait intervenir qu'un ou très peu d'individus par processeur. Les opérations génétiques telles que les croisements et les sélections n'ont lieu qu'entre individus voisins. Parfois, les meilleurs individus migrent de processeur en processeur. L'information est diffusée de proche en proche, ce qui vise à favoriser l'exploration. De nombreuses communications ont lieu et entraînent généralement des surcoûts de communications importants. La topologie utilisée est généralement une grille dans laquelle la majorité des individus ont quatre voisins. La Figure 2.13 présente un exemple de modèle cellulaire avec vingt individus qui ont entre deux et quatre voisins.

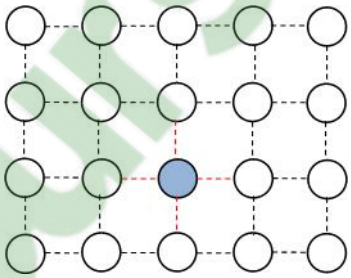


Figure 2.13: Modèle cellulaire pour les algorithmes évolutionnaires.

Comme le rapportent Luna et Alba (2015), très peu de travaux sont réalisés sur ce modèle car il requiert de nombreuses communications entre les processeurs. Ishibuchi *et al.* (2011) proposent une adaptation de l'algorithme de décomposition MOEA/D à un modèle cellulaire. Ainsi, chaque processeur est en charge d'un unique individu et optimise un problème uni-objectif obtenu par une fonction de somme pondérée. Des pondérations différentes sont affectées aux processeurs et doivent être préalablement définies. Les opérations de sélection et de remplacement sont effectuées en tenant compte des individus voisins. Même si l'expérimentation montre une qualité de solution jugée bonne par les auteurs, aucune information sur le temps d'exécution ni sur l'architecture utilisée n'est spécifiée. Finalement, le modèle par diffusion peut être adapté à des architectures spécifiques. Récemment, Li *et al.* (2013) proposent une parallélisation sur un processeur graphique (GPU) par le biais d'un modèle par diffusion.

2.6 SYNTHÈSE DES APPROCHES PARALLÈLES MULTI-OBJECTIFS

Cette section résume les modèles qui ont été présentés au cours de la revue de littérature en se concentrant sur le paradigme maître-esclave et le paradigme en îles qui constituent la majorité des travaux. Le Tableau 2.3 expose une vue d'ensemble des différentes approches maître-esclave en mettant en évidence les noms des auteurs, les algorithmes parallélisés, les types de problèmes traités, les nombres de coeurs de CPU ou GPU exploités et les accélérations maximales obtenues.

Auteurs	**Algorithme parallélisé**	**Problèmes traités**	**Cœurs**	**Acc.**
Durillo *et al.* (2008)	Version stationnaire de NSGA-II	Problème de réseau ad hoc	286	> 100
Ferringer *et al.* (2009)	ε-NSGA-II sur des clusters hétérogènes	Constellation de satellites	250	N/A
Mostaghim *et al.* (2008)	Essaim particulaire	Problèmes continus : ZDT	20	N/A
Depolli *et al.* (2013)	Évolution différentielle	Problèmes continus	28	>20
Lančinskas et Žilinskas (2012)	NSGA-II	Problème de localisation des installations	800	2048
Gupta et Tan (2015)	NSGA-II	Problèmes continus	GPU	10
Moreno *et al.* (2017)	NSGA-II	Problèmes continus DTLZ	GPU	15,91
Zhou et Tan (2009)	Essaim particulaire	Problèmes continus	GPU	11

Tableau 2.3: Caractéristiques des modèles maître-esclave présentés

En exploitant des cœurs de processeur CPU ou des GPUS, toutes les approches présentées permettent d'obtenir des accélérations significatives sur différents types de problèmes. Néanmoins, la majorité d'entre elles se focalisent sur des problèmes à variables continues et ne traitent pas des problèmes combinatoires. En outre, certaines approches utilisent des populations de taille très importante ce qui permet d'obtenir des gains de parallélisation plus élevés mais ne garantit pas l'obtention de solutions de meilleure qualité. Finalement, l'approche maître-esclave permet d'accélérer les métaheuristiques multi-objectifs sur des problèmes particuliers dont la phase d'évaluation est généralement très conséquente.

Le Tableau 2.4 permet de synthétiser les modèles en îles, qui ont été présentés dans les sections précédentes, en exposant notamment leur nom et leur particularité principale.

Nom	Auteurs	Particularité	Problèmes traités	Cœurs
DRMOGA	Hiroyasu *et al.* (2000)	Tri selon une fonction objectif	Problèmes continus	5
PSFGA	De Toro Negro *et al.* (2004)	Partitionnement de la population	Problèmes continus ZT	4
MFED	Essabri *et al.* (2006)	Échanges de fronts	Problèmes continus ZDT	2 à 10
PMRPEA	Figueira *et al.* (2010)	Points de référence	Problème de flowshop	10
MOGASOLS	Pilat et Neruda (2010)	Combinaison avec l'uni-objectif	Problèmes continus ZDT	2 à 9
PasMoQAP	Sanhueza *et al.* (2017)	Algorithme mémétique	Problème d'affectation quadratique	5 à 21
S-PAMICRO	Hernández Gómez *et al.* (2016)	Version parallèle de sms-emoa	Problèmes continus WFG	10 à 25
Séparation en cônes	Branke *et al.* (2004)	Basé sur NSGA-II	Problèmes continus ZDT	2 à 6
Modèle de regroupement	Streichert *et al.* (2005)	Algorithme de partitionnement	Problèmes continus	2 à 6
Basé sur la dominance guidée	Deb *et al.* (2003)	Modification de la dominance	Problèmes continus	2, 3 et 6
PSOV1, PSOV2	Bui *et al.* (2009)	Essaim particulaire et NSGA-II	Problèmes continus ZDT	5
CCMOEA	Dorronsoro *et al.* (2013)	Coévolution coopérative	RSMP	4 et 8
HybJacIsCone	Gourisaria *et al.* (2013)	Hybride avec NSGA-II	Problème du sac à dos	2 à 8
Hybrid model	Jagtap *et al.* (2011)	Hybride avec NSGA-II	Problème du sac à dos	8

Tableau 2.4: Caractéristiques des modèles en îles et hybrides présentés

Tout comme les modèles maître-esclave, la grande majorité des travaux basés sur le paradigme en îles concerne la résolution de problèmes à variables continues. De plus, bien que l'algorithme NSGA-II (Deb *et al.*, 2002) ait été proposé il y a plus de 15 ans, il reste encore l'objet de nombreux travaux de parallélisation alors que de nouveaux algorithmes évolutionnaires multi-objectifs ont été prouvés plus performants dans certains contextes. Ce tableau met également en évidence le faible nombre de processeurs généralement utilisé. En effet, la majorité des chercheurs se sont limités à six ou huit processeurs, ce qui peut

s'expliquer par le manque de disponibilité matérielle.

2.7 OBJECTIFS DE LA RECHERCHE

Quelques modèles de métaheuristiques parallèles ont été proposés dans la littérature pour répondre à des problèmes multi-objectifs. Cependant, bien qu'ils aient été prouvés performants dans leur contexte, ces modèles ont pour la plupart été expérimentés avec peu de cœurs de processeur et ne permettent généralement pas de profiter de la puissance de calcul des nouvelles architectures parallèles. D'une part, les approches maître-esclave sont fortement dépendantes du problème traité et leur extensibilité est souvent limitée à la durée de la phase d'évaluation. D'autre part, pour certains modèles en îles et avec un grand nombre de processeurs, la partie de l'espace objectif affectée à chaque processeur sera réduite et il sera alors difficile de trouver des solutions pour chaque île, notamment pour un problème many-objectives dans lequel beaucoup de solutions sont non-dominées. Pour exploiter les architectures de calcul haute performance, qui sont généralement des systèmes regroupant plusieurs types d'architectures, des modèles hybrides ont été proposés. Ces derniers hybrident un schéma maître-esclave qui permet de diminuer le temps de quelques opérations répétitives ainsi qu'un modèle en îles qui permet d'améliorer la qualité des solutions et accélérer ainsi les recherches. Les quelques publications concernant des modèles hybrides présentent des descriptions sommaires et des expérimentations incomplètes et ne permettent pas de conjecturer réellement sur les performances réelles des modèles proposés. De ce fait, de nombreuses perspectives de recherche existent dans ce domaine. Ainsi, l'objectif principal de cette thèse est de *contribuer à amener les modèles parallèles conçus pour les métaheuristiques multi-objectifs à exploiter efficacement les architectures de calcul haute performance*.

Trois objectifs spécifiques sont en corrélation avec l'objectif principal. Le premier objectif spécifique est de *proposer un nouveau modèle parallèle adapté aux algorithmes évolutionnaires multi-objectifs*. En s'inspirant des approches de la littérature et en prenant en compte leurs caractéristiques, il s'agit de proposer de nouvelles stratégies afin d'optimiser les performances au niveau du parallélisme tout en fournissant des solutions de meilleure qualité. Ces stratégies seront intégrées dans un modèle parallèle original qui visera l'exploitation efficace d'un nombre de processeurs plus important que les modèles existants.

Le deuxième objectif consiste à *établir une comparaison expérimentale des principaux modèles parallèles adaptés aux algorithmes évolutionnaires multi-objectifs*. Il n'existe pas

d'étude comparative récente concernant ces modèles, notamment pour mesurer leur extensibilité sur des architectures de calcul haute performance ou pour comparer leur convergence vers le front Pareto (Luna et Alba, 2015). Cette comparaison expérimentale permettra d'orienter l'utilisateur vers le modèle le plus adapté à ses besoins et à ses disponibilités matérielles. Pour ce faire, les expérimentations seront basées sur l'algorithme GISMOO (Zinflou *et al.*, 2012) qui est relativement récent et qui s'est montré compétitif par rapport à des algorithmes tels que NSGA-II ou SPEA-II que ce soit en termes de convergence vers le front Pareto ou de diversité des solutions. De plus, à notre connaissance, aucun travail concernant sa parallélisation n'a été effectué.

Enfin le dernier objectif spécifique vise à *résoudre efficacement des problèmes combinatoires multi-objectifs à travers le modèle proposé et analyser les performances*. Ce type de problème a été peu traité comparativement aux problèmes à variables continues dans le domaine de l'optimisation parallèle multi-objectifs (Section 2.6). Les résultats du modèle parallèle devront se montrer compétitifs par rapport aux résultats des approches de la littérature autant au niveau du temps d'exécution que de la qualité des solutions obtenues. Le supercalculateur ROMEO (ROMEO, 2018), basé à Reims, permettra d'exécuter les différents modèles. En plus de sa puissance de calcul importante, ROMEO hybride des architectures qui sont très utilisées dans le domaine du calcul HPC puisqu'il possède de nombreux nœuds de calcul distribués qui ont chacun une architecture à mémoire partagée et multicœur.

CHAPITRE 3

PROPOSITION D'UN MODÈLE PARALLÈLE ASYNCHRONE POUR LES ALGORITHMES ÉVOLUTIONNAIRES MULTI-OBJECTIFS

Dans le chapitre précédent, les principaux modèles parallèles pour les AEMO proposés dans la littérature ont été présentés. Bien que ces modèles parallèles aient apporté une contribution significative à l'amélioration de certains AEMO, ils ont souvent été conçus pour des méthodes et problèmes spécifiques. De ce fait, leur généricité et leur performance dans une variété de contextes ont rarement été étudiées. De même, ces modèles ont souvent des difficultés à atteindre une bonne approximation du front Pareto tout en présentant une extensibilité importante. Dans ce chapitre, un nouveau modèle en îles nommé APM-MOEA (*Asynchronous Parallel Model for Multi-objective Evolutionary Algorithms*) est proposé pour répondre aux limites des modèles existants. La proposition du modèle a fait l'objet d'une publication pour la conférence internationale *7th International Conference on Metaheuristics and Nature Inspired Computing (META'18)* (Mazière *et al.*, 2018).

Dans l'approche APM-MOEA, la répartition de l'approximation du front Pareto est basée sur une stratégie *diviser pour régner* telle que proposée dans le modèle de Streichert *et al.* (2005). Une des principales limitations de ce dernier, rapporté par Jaimes et Coello (2005), est que le groupement des sous-populations à intervalles réguliers peut entraîner des surcoûts de communication importants qui augmentent avec le nombre d'îles utilisées. Par ailleurs, à notre connaissance, aucun travail concernant l'optimisation de problèmes combinatoires avec ce modèle n'est disponible dans la littérature. Néanmoins, l'adaptation du modèle pour GISMOO (Zinflou *et al.*, 2012) et la phase d'expérimentation ont révélé les performances de celui-ci dans la résolution d'instances de problèmes du voyageur de commerce multi-objectifs avec un faible nombre d'îles. Ces travaux, présentés à la conférence nationale Compas'16 (Mazière

et al., 2016b), ont mis en évidence les difficultés du modèle dans l'exploitation de 8 ou 16 processeurs et dans la résolution d'instances dont les fronts Pareto sont pauvres en nombre de solutions. Ainsi, la motivation de cette recherche est de proposer un nouveau modèle parallèle pour les AEMO qui suit les principes de découpage de l'espace objectif tels que définis par Streichert *et al.* (2005) tout en répondant aux limites de cette approche.

Dans la première partie de ce chapitre, l'approche *diviser pour régner* utilisée dans le modèle APM-MOEA est décrite avec la proposition de mécanismes pour améliorer la répartition des solutions sur les différentes îles. Les nouvelles composantes et caractéristiques du modèle en îles sont ensuite exposées. Elles permettent d'améliorer la convergence et la diversification des AEMO, mais aussi de limiter les surcoûts du parallélisme. Le comportement global du modèle est détaillé par le biais d'un schéma de fonctionnement et de plusieurs pseudo-codes dans la troisième partie.

3.1 STRATÉGIE *DIVISER POUR RÉGNER*

Les algorithmes évolutionnaires étudiés dans cette thèse ont pour but de fournir de multiples solutions de compromis en une seule exécution. La recherche de ces solutions peut se faire de manière relativement indépendante et donc simultanément en exploitant différentes unités de calcul. À partir de ce constat, des approches parallèles basées sur une stratégie *diviser pour régner* ont été proposées et se sont révélées efficaces pour des algorithmes évolutionnaires (Branke *et al.*, 2004) (Streichert *et al.*, 2005) et des algorithmes par essaim particulaire (Liu *et al.*, 2018) dans la résolution de problèmes continus multi-objectifs. En reprenant les principes du paradigme en îles, ces modèles découpent explicitement l'espace objectif (ou l'espace de recherche) en différentes régions et attribuent chaque sous-région à une île particulière. Chaque île est alors en charge de trouver des solutions dans une zone spécifique de l'espace objectif. Le nouveau modèle proposé dans ce chapitre suit le modèle de Streichert *et al.* (2005) en utilisant un algorithme de partitionnement pour subdiviser l'espace objectif.

3.1.1 MODÈLE DE Streichert et al. (2005)

Le modèle en îles de Streichert *et al.* (2005), désigné *modèle de clustering* par la suite, a été prouvé efficace sur plusieurs problèmes à variables continues. Celui-ci est notamment plus flexible quant à la forme du front Pareto que la méthode de séparation en cônes (Branke *et al.*,

2004). Le comportement global de ce modèle suit le paradigme en îles et reprend les principes de *diviser pour régner* pour paralléliser un AEMO. Un algorithme de partitionnement est utilisé pour répartir l'approximation du front Pareto sur les cœurs de processeur disponibles. Sur chaque île i, l'algorithme NSGA-II (Deb *et al.*, 2002) permet de faire évoluer une sous-population P_i. À intervalles réguliers, toutes les sous-populations sont rassemblées, regroupées grâce à l'algorithme de partitionnement et redistribuées sur les îles correspondantes. Le pseudo-code du *modèle de clustering* est détaillé à l'Algorithme 2. Dans un premier temps, les sous-populations P_i sont initialisées et évaluées sur chaque île i (lignes 1-4). Un processus itératif est ensuite répété jusqu'à atteindre un critère d'arrêt défini. Dans celui-ci, les sous-populations évoluent à travers les opérations de l'AEMO parallélisé (lignes 6-10). Puis, à intervalles réguliers, les sous-populations sont regroupées (lignes 12-14) sur une île particulière qui est en charge d'appliquer l'algorithme de regroupement (ligne 15). Les résultats de ce dernier sont envoyés aux autres îles et permettent de mettre à jour leur zone de recherche (ligne 16-17).

Algorithme 2: Pseudo-code du modèle de Streichert *et al.* (2005)

```
1  Pour chaque sous-population P_i Faire
2      Initialiser P_i ;
3      Évaluer P_i ;
4  FinPour
5  Tant que condition d'arrêt non atteinte Faire
6      Pour chaque sous-population P_i Faire
7          Faire évoluer P_i;
8          Évaluer P_i ;
9      FinPour
10     si point de migration atteint alors
11         Initialiser population Q = ∅;
12         Pour chaque sous-population P_i Faire
13             Ajouter P_i à Q;
14         FinPour
15         Effectuer le regroupement sur Q;
16         Mettre à jour les P_i;
17         Mettre à jour les contraintes;
18     finsi
19 FinTantque
```

3.1.2 ALGORITHME DE PARTITIONNEMENT

Partitionnement en *k*-moyennes (*k-means*) La méthode de partitionnement utilisée dans le *modèle de clustering* est l'algorithme de partitionnement en *k*-moyennes qui a été introduit par Steinhaus (1956) et qui est l'un des algorithmes de regroupement les plus populaires dans la littérature. Dans sa dénomination, *k* correspond au nombre de groupes (*clusters*) désirés, c'est-à-dire au nombre d'îles dans le modèle parallèle. L'Algorithme 3 présente le pseudo-code de l'approche. Après une initialisation aléatoire des centroïdes [1], deux étapes sont répétées jusqu'à obtenir un état de stabilité : une phase d'affectation des points au groupe le plus proche et une phase dans laquelle les centroïdes sont mis à jour.

Algorithme 3: Pseudo-code du partitionnement *k*-moyennes

Données: *k* : Le nombre de groupes,
E : Un ensemble de points.
Retour: Un ensemble de *k* groupes

```
1 Choisir aléatoirement k centroïdes parmi E;
2 Tant que aucune affectation de groupe ne change Faire
3 |   Assigner chaque point de E au groupe dont le centroïde est le plus proche;
4 |   Mettre à jour les k centroïdes en calculant la moyenne des points de chaque groupe;
5 FinTantque
```

Amélioration de la méthode *k*-moyennes Les résultats de ce partitionnement dépendent fortement des centroïdes initiaux qui sont choisis aléatoirement ce qui constitue le principal inconvénient de l'approche puisque l'aspect stochastique n'assure pas la robustesse des résultats. Plusieurs méthodes ont été proposées pour améliorer l'efficacité du partitionnement en *k*-moyennes et notamment l'initialisation des centroïdes. Parmi elles, Yedla *et al.* (2010) ont introduit un nouvel algorithme de partitionnement qui inclut une méthode pour définir de meilleurs points initiaux et une approche plus efficace pour assigner les points à leur groupe correspondant. Les expérimentations ont montré que leur approche est plus performante et robuste, mais aussi plus rapide que l'algorithme de *k*-moyennes original. Les grandes lignes de cette approche sont intégrées dans APM-MOEA et le point d'origine, qui est utilisé pour initialiser les centroïdes, est remplacé par le point idéal pour prendre en compte les caractéristiques de la recherche dans l'espace objectif.

1. Centre d'un groupe, moyenne de tous les points associés au groupe

Un algorithme de partitionnement peut parfois amener à des groupes vides et donc des centroïdes nuls puisqu'ils ne seraient associés à aucun point. Dans le modèle APM-MOEA, il est indispensable d'éviter ce genre de situation car une île serait en charge de trouver des solutions près d'un point nul dans l'espace objectif. Pour prévenir des groupes vides, une méthode a été proposée par Pakhira (2009). Le calcul pour mettre à jour un centroïde à chaque itération est modifié puisqu'il dépend désormais des anciennes cordonnées de celui-ci. L'Équation 3.1 définit mathématiquement la mise à jour du k-ième centroïde Z_k.

$$Z_k = \frac{1}{n_k + 1} \left\{ \sum_{x_j \in C_k} (x_j) + z_{k(old)} \right\} \tag{3.1}$$

où C_k est la partition k, n_k est le nombre de points associés au groupe C_k et $z_{k(old)}$ correspond aux anciennes coordonnée de Z_k.

D'après leur phase d'expérimentation, la nouvelle approche proposée permet de prévenir les groupes vides tout en conservant des performances similaires à l'algorithme de partitionnement k-moyennes.

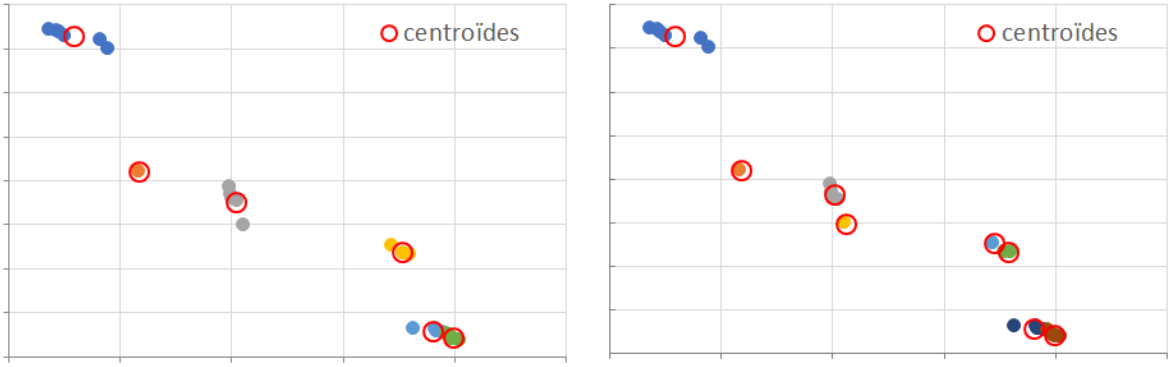
Proposition d'un nouvel algorithme de partitionnement Finalement, en s'appuyant sur l'initialisation améliorée de Yedla *et al.* (2010) et sur la méthode de mise à jour des centroïdes de Pakhira (2009), une nouvelle version d'un algorithme de partitionnement est proposée pour APM-MOEA et le pseudo-code est présenté à l'Algorithme 4. Le partitionnement d'une archive de vecteurs objectif en k groupes débute par le calcul du point idéal (ligne 1). Ensuite, un premier partitionnement équilibré est effectué en fonction des distances des vecteurs objectif au point idéal (lignes $2-4$). Les centroïdes initiaux sont calculés pour chaque groupe en considérant les moyennes des vecteurs (ligne 5). Les prochaines étapes sont répétées jusqu'à atteindre un état de stabilité dans lequel aucune nouvelle assignation n'est effectuée. Chaque vecteur objectif v_i est d'abord réassigné au groupe dont le centroïde est le plus proche (lignes $8-12$). Les centroïdes sont ensuite mis à jour selon l'Équation 3.1 (ligne 14). Pour chaque vecteur, il faut vérifier si le centroïde qui lui est le plus proche est celui du groupe auquel il est assigné. Si tel est le cas, le vecteur restera dans le même groupe, sinon les distances aux centroïdes sont recalculées et il devra être réaffecté au groupe dont le centroïde est le plus proche. Si au moins un vecteur doit être réassigné à un autre groupe, alors le processus devra être réitéré.

Algorithme 4: Pseudo-code du partitionnement en k groupes

Données: $V = \{v_1, v_2, ..., v_n\}$: Archive de vecteurs objectif,
k : Nombre de groupes.
Retour: Un ensemble de k groupes et leur centroïdes

Définir le vecteur idéal à partir de l'archive V;
Pour chaque vecteur objectif, calculer la distance au vecteur idéal;
Trier l'archive V à partir des distances précédemment calculées;
Partitionner l'archive V triée en k groupes égaux;
Définir les centroïdes de chaque groupe;
Calculer les distances entre les centroïdes et vecteurs.
Tant que *aucun changement de groupe n'est constaté* **Faire**
    Pour *chaque vecteur* $v_i \in V$ **Faire**
        Trouver le centroïde c_j le plus proche de v_i;
        $DistCent[i]$ = distance entre c_j et v_i;
        Assigner v_i au groupe j
    FinPour
    Mettre à jour les centroïdes avec L'Équation 3.1;
    Pour *chaque vecteur* $v_i \in V$ **Faire**
        Calculer la distance d_i entre v_i et le centroïde du groupe assigné;
        si $d_i > DistCent[i]$ **alors**
            Calculer les distances entre tous les centroïdes et v_i;
        sinon
            v_i reste dans le même groupe
        finsi
    FinPour
FinTantque

Afin d'illustrer les avantages de la nouvelle méthode, la Figure 3.1 présente un exemple de partitionnement en huit groupes pour l'algorithme de *k*-moyennes (3.1a) et l'algorithme de partitionnement proposé (3.1b). Les points de données correspondent à des vecteurs objectif obtenus lors de la résolution d'un problème bi-objectifs. Pour la méthode de *k*-moyennes originale, seulement six groupes sont exposés puisque les deux groupes manquants sont vides et sont associés à des centroïdes nuls. Le partitionnement proposé fournit, quant à lui, huit groupes contenant au moins un vecteur et des centroïdes non nuls. Cet exemple expose les problèmes du partitionnement en *k*-moyennes sur des ensembles de vecteurs discontinus et l'avantage de la méthode de Pakhira (2009) pour éviter les groupes vides.



(a) Partitionnement en *k*-moyennes **(b) Partitionnement amélioré**

Figure 3.1: Exemples de partitionnement avec huit groupes

3.1.3 RESTRICTION DES ÎLES À LEUR RÉGION

Les centroïdes calculés lors du partitionnement permettent de limiter les îles à des régions spécifiques de l'espace objectif. Pour ce faire, des contraintes sont ajoutées au problème. Pour chaque île *i*, une solution *S* est marquée comme réalisable si et seulement si le plus proche centroïde de S est celui associé à l'île *i*. Dans le cas contraire, la solution *S* est considérée comme non réalisable et n'est pas favorisée lors de la sélection et du remplacement de l'AEMO. Néanmoins, si la solution est non-dominée, elle sera envoyée indirectement à l'île correspondante (Section 3.1.4). Afin de différencier les solutions non réalisables, un degré de violation est calculé pour chacune d'entre elles et correspond à la distance euclidienne entre le vecteur objectif et le centroïde de l'île considérée. La Figure 3.2 expose un exemple d'un espace objectif avec quatre centroïdes et des solutions obtenues par l'île associée au centroïde 2. Comme le montre l'exemple, les solutions les plus proches du centroïde 2 sont des solutions

réalisables et les autres sont considérées non réalisables.

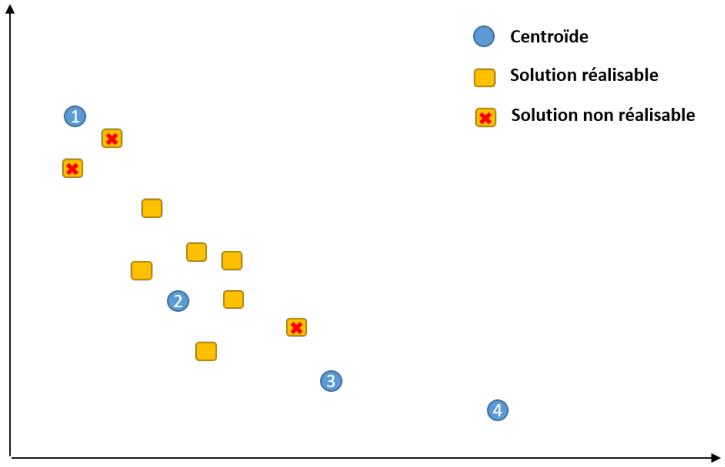


Figure 3.2: Exemple de solutions réalisables et infaisables avec quatre groupes

Le principe de dominance sous contraintes introduit par Deb *et al.* (2002) modifie la définition de la dominance Pareto en intégrant la faisabilité des solutions et permet de comparer les solutions dans le problème multi-objectifs contraint. Ainsi, une solution X domine sous contrainte une solution Y si :

— X est réalisable et Y est non réalisable.

— X et Y sont réalisables et $X \prec Y$.

— X et Y sont non réalisables et X a un degré de violation inférieur à Y.

La dominance sous contrainte est utilisée pour comparer les solutions lors de la phase de sélection de l'AEMO et pour trier les solutions dans l'opération de remplacement.

3.1.4 VUE GLOBALE DE L'ÎLE ORGANISATRICE

Une approche *diviser pour régner* a besoin d'une entité désignée qui dirige l'ensemble des processus. Dans le modèle APM-MOEA, une île, nommée île organisatrice, est en charge de la répartition du travail sur les autres îles et possède une vue globale de l'état courant de la recherche. Pour ce faire, une archive globale est continuellement mise à jour en intégrant les archives locales de toutes les autres îles. Ce choix d'utiliser une archive globale de solutions non-dominées, au détriment des sous-populations à la manière du *modèle de clustering*, est motivé par les réussites des approches élitistes pour les AEMO comme souligné dans la Section 1.3.3. La dominance au sens Pareto est utilisée pour intégrer les nouvelles solutions

dans l'archive globale et dans les archives locales. Les contraintes sur les zones de l'espace objectif ne sont pas prises en compte pour ces archives. À la place d'envoyer les archives locales complètes, les autres îles envoient seulement les nouvelles solutions non-dominées trouvées. Pour ce faire, une archive annexe est utilisée et est mise à jour en fonction des nouvelles solutions ajoutées à l'archive locale. Celle-ci est réinitialisée après chaque envoi à l'île organisatrice. L'Algorithme 5 présente le pseudo-code de la mise à jour de l'archive locale *archLocale* et de l'archive *archAnnexe* en fonction d'une population déjà évaluée *pop*. La fonction $miseAjour(x, arch)$ permet d'ajouter la solution x à l'archive *arch* si x n'est dominée par aucune solution de *arch*. Elle renvoie un booléen indiquant si la solution x a été ajoutée.

Algorithme 5: Pseudo-code de mise à jour des archives

```
1 Fonction majArchives(Population pop, Archive archLocale, Archive archAnnexe)
2     Pour chaque individu x ∈ Pop Faire
3         estAjoute = miseAjour(x, archLocale);
4         si estAjoute alors
5             miseAjour(x, archAnnexe);
6         finsi
7     FinPour
```

Il faut noter que si l'AEMO parallélisé ne maintient pas d'archive de solutions non-dominées ou si les solutions de l'archive ne sont pas utilisées dans les opérations évolutionnaires, alors il n'est pas nécessaire d'affecter une archive locale par île. Dans ce cas, seule une archive annexe, qui est également réinitialisée après chaque envoi, est mise à jour sur chaque île.

3.2 COMPOSANTES ET CARACTÉRISTIQUES DU MODÈLE

Dans la section précédente, la répartition de la recherche du front Pareto sur les différentes îles a été décrite pour le modèle APM-MOEA. Les composantes et caractéristiques additionnelles du modèle parallèle vont maintenant être présentées. Elles visent à répondre à des objectifs d'amélioration de qualité de solutions ou des objectifs d'optimisation du parallélisme.

3.2.1 AMÉLIORATION LOCALE

La plupart des travaux qui se concentrent sur la résolution de problèmes combinatoires multi-objectifs utilisent une quelconque forme de recherche locale dans leur processus de recherche, soit avec l'hybridation d'un AEMO ou dans un algorithme de recherche locale Pareto autonome. Dans le modèle proposé APM-MOEA, une recherche locale non-itérative est appliquée sur chaque solution de la population à intervalles réguliers, c'est-à-dire toutes les *nGen* générations. À la place d'explorer de manière exhaustive le voisinage d'une solution, une exploration partielle est favorisée et permet de réduire les temps de calcul. En fonction des caractéristiques du problème traité, deux stratégies peuvent être adoptées pour effectuer cette exploration : la stratégie *first improvement* et la stratégie avec liste de candidats. Dans la stratégie $firstimprovement$, décrite à l'Algorithme 6, tous les voisins s' d'une solution s sont générés jusqu'à obtenir une meilleure solution, c'est-à-dire $s' \prec s$ (s' domine au sens Pareto s). Dans la seconde stratégie, dont le pseudo-code est présenté à l'Algorithme 7, les voisins générés sont limités à une liste de candidats L qui doivent être définis par rapport au problème à résoudre. Tous les voisins qui dominent la solution courante sont ajoutés dans une nouvelle archive A. À la fin du processus, une solution non-dominée est choisie aléatoirement dans l'archive et remplace la solution de la population.

Algorithme 6: Amélioration locale : stratégie *first improvement*

```
   Données: Population P
1  Pour chaque s ∈ P Faire
2  |  s' = voisin de s;
3  |  Tant que s' ⊀ s Faire
4  |  |  s' = voisin de s;
5  |  FinTantque
6  |  s = s';
7  FinPour
```

Algorithme 7: Amélioration locale : stratégie avec liste de candidats

```
   Données: Population P
1  Archive a;
2  Pour chaque s ∈ P Faire
3  |  A = ∅;
4  |  L = voisins candidats de s;
5  |  Pour chaque l ∈ L Faire
6  |  |  si l ≺ s alors
7  |  |  |  miseAjour(l, A);
8  |  |  finsi
9  |  FinPour
10 |  r = entier aléatoire entre 1 et |A|;
11 |  s = a[r];
12 FinPour
```

Les solutions de problèmes combinatoires sont souvent représentées comme une permu-

tation d'éléments. Pour générer le voisinage des solutions avec ce type de représentation, plusieurs méthodes de perturbation existent par exemple : l'opérateur d'échange $1-Opt$ (*swap mutation*), la mutation par inversion ou encore l'opérateur $2-Opt$. En passant d'une solution s à une solution voisine s' par le biais d'une de ces méthodes, il est possible que le calcul des fonctions objectif de s' se fassent rapidement et facilement à partir de celles de s. En effet, dans l'amélioration locale de APM-MOEA et pour certains problèmes multi-objectifs, il n'y a pas besoin de calculer les fonctions objectif des nouveaux voisins générées à partir de zéro.

3.2.2 AJOUT D'ÎLES TÉMOINS

Pour améliorer la capacité d'exploration du modèle, des recherches en dehors de l'approche *diviser pour régner* peuvent être effectuées. Pour ce faire, deux îles supplémentaires, nommées îles témoins, sont incluses dans le modèle APM-MOEA afin de trouver des solutions mieux diversifiées et d'explorer l'espace objectif de manière plus efficace. Contrairement aux autres îles, elles n'ont pas de contrainte sur leur zone de recherche et doivent trouver des solutions dans tout l'espace objectif. La première île témoin utilise la recherche locale pour améliorer les solutions trouvées alors que la seconde créée de nouveaux individus uniquement avec des opérations de l'AEMO. Elles communiquent exclusivement et de manière asynchrone avec l'île organisatrice qui fournit de nouvelles solutions provenant de l'archive globale. Cela permet de donner une vue globale de la recherche actuelle aux deux îles et ainsi explorer l'ensemble de l'espace objectif. À l'instar des autres îles, elles envoient périodiquement leur archive annexe à l'île organisatrice.

3.2.3 ÉCHANGES DE COMMUNICATION ASYNCHRONES

Comme il a été rapporté par Jaimes et Coello (2005), les échanges de population à intervalles réguliers dans le modèle de Streichert *et al.* (2005) peuvent engendrer des surcoûts de communication importants. Pour répondre à ce problème et gérer l'irrégularité des calculs de la recherche locale, le modèle APM-MOEA utilise des communications asynchrones ce qui permet d'éviter l'ajout de barrières de synchronisation globales entre les processeurs. Grâce à l'asynchronisme, l'île organisatrice n'a pas besoin d'attendre toutes les archives annexes des îles avant d'effectuer un nouveau partitionnement. À intervalles réguliers, elle vérifie

simplement si de nouvelles archives sont en cours de réception ou sont effectivement reçues et met à jour en conséquence l'archive globale. Toutes les opérations d'envoi sont non-bloquantes et n'ont donc pas besoin d'attendre les opérations de réception correspondantes. Les coûts du parallélisme pour les autres îles sont également limités car elles n'ont qu'à périodiquement envoyer leur archive annexe et vérifier la réception de nouveaux groupes de solutions provenant de l'île organisatrice. Du point de vue de l'implémentation, les données envoyées ne doivent être, en aucun cas, modifiées avant la réception effective de celles-ci. La Figure 3.3 expose les différences entre des communications synchrones et des communications asynchrones dans le modèle APM-MOEA avec seulement une île organisatrice et une île classique. Elle témoigne de la diminution voire de la suppression des temps d'attente dans les processus de recherche quand des communications asynchrones sont effectuées. Les périodes d'attente sont remplacées par des vérifications périodiques qui permettent de diminuer les surcoûts liés au modèle parallèle.

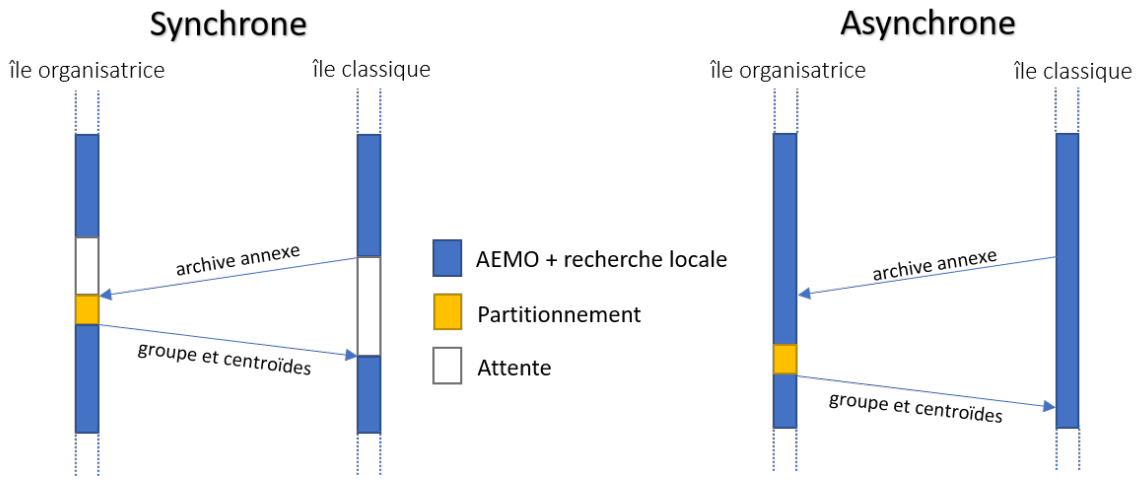


Figure 3.3: Schéma de communications synchrones et asynchrones

3.3 SCHÉMA GLOBAL DE APM-MOEA

L'approche APM-MOEA est un modèle parallèle dans lequel plusieurs processus, nommés îles, coopèrent pour optimiser de manière efficace un problème multi-objectifs. Une île organisatrice et plusieurs autres îles (îles classiques) font évoluer des populations à travers un AEMO prédéfini et une amélioration locale pour trouver des solutions dans une partie spécifique de l'espace objectif. En même temps, deux îles témoins évoluent également des

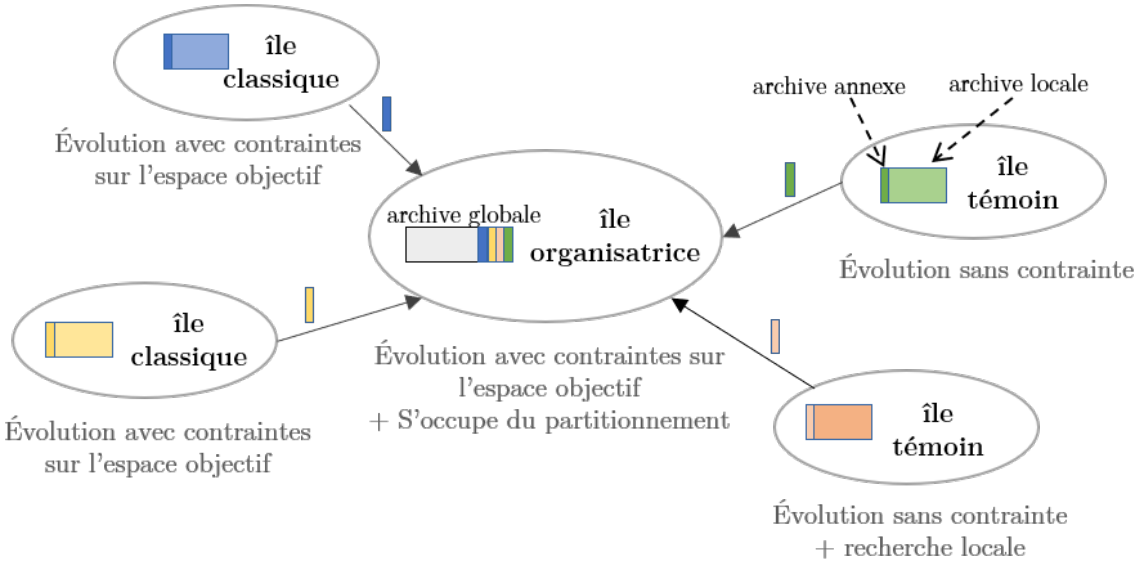


Figure 3.4: Schéma globale des communications vers l'île organisatrice

populations mais sans contrainte sur les zones de recherche. En coopération avec les îles classiques et les îles témoins, l'île organisatrice maintient une archive globale de solutions non-dominées et est en charge de définir les régions de l'espace objectif attribuées aux îles classiques. La Figure 3.4 permet de résumer le fonctionnement de APM-MOEA en indiquant les communications vers l'île organisatrice et en précisant le rôle de chaque île au sein du modèle. Les envois asynchrones des archives annexes permettent de mettre à jour l'archive globale afin que l'île organisatrice puisse effectuer un nouveau partitionnement et calculer les nouvelles zones attribuées à chaque île classique. Une fois le partitionnement terminé, les groupes de solutions et les centroïdes sont envoyés aux îles classiques. De même, l'archive globale complète est partagée avec les îles témoins.

Le fonctionnement de l'île organisatrice est détaillé à l'Algorithme 8. Les premières étapes (lignes 1-5) permettent d'initialiser les ensembles de solutions, mais également les centroïdes utilisés pour restreindre les îles à leur région. Un processus itératif est ensuite répété jusqu'à atteindre un certain critère d'arrêt. Ce dernier peut être un nombre de générations, un nombre d'évaluations ou encore un temps d'exécution maximal. À chaque itération, la population de l'île évolue pendant *nGen* générations grâce aux opérations évolutionnaires de l'AEMO et les archives sont continuellement mises à jour avec les nouvelles solutions générées. La recherche se concentre sur la zone de l'espace objectif représentée par le centroïde n^o 1 (ligne 8). Une fois l'évolution terminée, les solutions de la population sont améliorées localement

(ligne 11) par l'une des deux méthodes présentées à la Section 3.2.1. L'archive globale est ensuite éventuellement mise à jour en fonction des nouvelles archives annexes reçues depuis les autres îles (lignes 13-16). L'étape suivante est effectuée si le prochain point de migration est atteint et si de nouvelles archives ont été reçues précédemment. Elle commence par un partitionnement de l'archive globale en $k-2$ groupes où k est le nombre total d'îles. Chaque groupe est envoyé à son île classique correspondante et les centroïdes sont diffusés à toutes les îles classiques (lignes 19-22). Par ailleurs, l'archive globale complète est envoyée aux deux îles témoins (lignes 23-25). Une fois tous les envois asynchrones effectués, le premier groupe obtenu par le partitionnement est intégré à la population. Les centroïdes, qui sont utilisés pour contraindre le problème, sont également mis à jour grâce au résultat du partitionnement. Les différentes étapes présentées sont réitérées jusqu'à atteindre la condition d'arrêt et la fin de la recherche est signalée aux autres îles.

Algorithme 8: Pseudo-code détaillé de l'île organisatrice

```
   Données: k, nombre d'îles
1  Archive archLocale = ∅, archGlobale = ∅;
2  Initialisation aléatoire de la population Pop;
3  miseAjour(Pop, archLocale);
4  Booléen estModifie = faux;
5  Initialiser aléatoirement les centroïdes;
6  Tant que condition d'arrêt non atteinte Faire
7      Pour g allant de 1 à nGen Faire
8          Faire évoluer Pop avec l'AEMO autour du centroïde n° 1;
9          miseAjour(Pop, archLocale);
10     FinPour
11     Amélioration locale de Pop;
12     si de nouvelles archives annexes sont reçues alors
13         Mettre à jour archGlobale avec les archives annexes;
14         estModifie = vrai;
15     finsi
16     si point de migration atteint alors
17         si estModifie == vrai alors
18             Partitionner archGlobale en k − 2 groupes;
19             Pour i allant de 1 à k − 3 Faire
20                 Envoyer le groupe i + 1 à l'île classique i;
21                 Envoyer les centroïdes à l'île classique i;
22             FinPour
23             Pour it allant de 1 à 2 Faire
24                 Envoyer archGlobale à l'île témoin it;
25             FinPour
26             Intégrer le groupe 1 dans Pop;
27             Mettre à jour les contraintes avec les nouveaux centroïdes;
28             estModifie = faux
29         finsi
30     finsi
31 FinTantque
32 Signaler l'arrêt aux autres îles;
33 retourner archGlobale;
```

L'Algorithme 9 présente le pseudo-code d'une île classique. Les étapes d'initialisation

(lignes 1-4) et d'évolution des populations (lignes 6-10) sont similaires à celles de l'île organisatrice. Cette fois-ci, l'île classique n_i est chargée de trouver des solutions proches du centroïde n_i. À chaque itération, si de nouvelles données sont reçues, le nouveau groupe est intégré et les contraintes sont mises à jour avec les centroïdes (lignes 11-14). À chaque étape de migration, les archives annexes sont d'abord envoyées à l'île organisatrice puis remises à zéro (lignes 15-18). Une île classique s'arrête lorsque l'île organisatrice lui a signalé la fin de la recherche. En outre, les îles témoins ont un comportement similaire, mais ne sont pas limitées à des zones spécifiques de l'espace de recherche et réceptionnent les solutions de l'archive globale.

Algorithme 9: Pseudo-code détaillé d'une île classique

```
   Données: n_i, numéro de l'île classique
1  Archive archLocale = ∅, archAnnexe = ∅;
2  Initialisation aléatoire de la population Pop;
3  Initialiser aléatoirement les centroïdes;
4  miseAjour(Pop, archLocale);
5  Tant que l'arrêt n'est pas signalé par l'île organisatrice Faire
6      Pour g allant de 1 à nGen Faire
7          Faire évoluer Pop avec l'AEMO autour du centroïde n_i;
8          majArchives(Pop, archLocale, archAnnexe);
9      FinPour
10     Amélioration locale de Pop;
11     si un groupe et des nouveaux centroïdes sont reçus alors
12         Intégrer le groupe dans Pop;
13         Mettre à jour les contraintes avec les centroïdes reçues;
14     finsi
15     si point de migration atteint alors
16         Envoyer archAnnexe à l'île organisatrice;
17         archAnnexe = ∅;
18     finsi
19 FinTantque
```

3.4 CONCLUSION

L'approche parallèle APM-MOEA, qui a pour objectif l'amélioration et l'accélération d'un AEMO, a été proposée au cours de ce chapitre. Elle est basée sur le paradigme en îles et utilise un nouvel algorithme de partitionnement pour répartir la recherche du front Pareto sur les différentes îles. Chaque île exécute un AEMO et maintient une archive locale contenant l'ensemble des solutions non-dominées. Différentes îles sont utilisées dans le modèle : une île organisatrice, plusieurs îles classiques et des îles témoins. Comme son nom le suggère, l'île organisatrice joue un rôle central dans le modèle puisqu'elle communique de manière bidirectionnelle et exclusive avec toutes les autres îles. Les principales caractéristiques du modèle et les contributions pertinentes ont également été détaillées afin de mieux comprendre

son fonctionnement. D'une part, la vue globale donnée à l'île organisatrice ainsi que le nouvel algorithme de partitionnement visent à mieux répartir la recherche du front Pareto sur les îles. D'autre part, les îles témoins et l'algorithme de recherche locale non-itératif ont pour but d'améliorer la qualité des solutions autant au niveau de la diversité que de la convergence. Enfin, les communications asynchrones répondent à des optimisations de parallélisme, ce qui permet d'envisager la réduction du temps de résolution.

Afin de valider scientifiquement l'approche, le prochain chapitre est consacré à l'étude expérimentale de certaines composantes de APM-MOEA sur les bases d'une comparaison avec les modèles parallèles de la littérature dans un environnement d'exécution commun.

CHAPITRE 4

ÉTUDE COMPARATIVE DE MODÈLES DISTRIBUÉS POUR LA PARALLÉLISATION DE L'ALGORITHME MULTI-OBJECTIFS GISMOO

De multiples approches parallèles ont été proposées dans la littérature pour accélérer et améliorer les AEMO. Celles-ci ont généralement été conçues et expérimentées pour des algorithmes spécifiques et des problèmes particuliers. De même, leur extensibilité est généralement éprouvée avec seulement deux ou quatre cœurs de processeur. Leur généricité et leur performance dans une variété de contextes ont ainsi rarement été étudiées. De ce fait, comme il n'existe pas, à notre connaissance, d'étude comparative récente des principaux modèles parallèles existants, ce chapitre est consacré à la comparaison empirique de plusieurs approches dans la parallélisation de l'algorithme multi-objectifs GISMOO (Zinflou *et al.*, 2012). Ces travaux ont, par ailleurs, été présentés aux conférences nationales COMPAS'16 (*Conférence d'informatique en Parallélisme, Architecture et Système, Lorient*) (Mazière *et al.*, 2016b) et ROADEF 2016 (*Congrès annuel de la société Française de Recherche Opérationnelle et d'Aide à la Décision, Compiègne*) (Mazière *et al.*, 2016a). En plus de fournir une comparaison des modèles parallèles existants, cette étude permettra d'analyser certaines composantes du modèle APM-MOEA qui a été présenté dans le chapitre précédent.

Dans la première partie du chapitre, les éléments de référence sur lesquels seront basées les expérimentations sont précisés. Les deux problèmes combinatoires qui seront traités tout au long de cette thèse sont décrits et les principales instances de test sont énumérées. De même, l'algorithme GISMOO, qui servira de référence pour les multiples modèles parallèles comparés, est détaillé. Sa compétitivité par rapport à des AEMO classiques est mise en évidence en utilisant un cadre d'applications nommé ParadisEO (Cahon *et al.*, 2004). Finalement, une première comparaison empirique d'une version de APM-MOEA et des principaux modèles

en îles, adaptés à l'algorithme GISMOO, est effectuée à la fin de ce chapitre. Elle permet de comparer à la fois les modèles parallèles de la littérature et d'analyser certaines contributions spécifiques du modèle APM-MOEA, ce qui constitue les prémisses de travaux concernant une expérimentation complète du nouveau modèle proposé.

4.1 ÉLÉMENTS DE RÉFÉRENCE

Pour expérimenter les modèles parallèles, plusieurs éléments de référence ont été choisis. Ainsi, la suite de ce chapitre décrit les problèmes multi-objectifs à optimiser et l'AEMO qui sera accéléré et amélioré.

4.1.1 PROBLÈMES COMBINATOIRES TRAITÉS

Cette thèse s'intéresse essentiellement aux problèmes combinatoires qui représentent une part importante des problèmes réels rencontrés dans des domaines variés tels que la biologie, l'ingénierie ou encore la médecine (Munakata et Barták, 2010). Les deux problèmes à variables discrètes qui seront optimisés pendant les multiples expérimentations de cette recherche sont présentés par la suite.

4.1.1.1 LE PROBLÈME D'AFFECTATION QUADRATIQUE MULTI-OBJECTIFS

Formulation du problème de MQAP Le problème d'affectation quadratique (**QAP** : *Quadratic Assignment Problem*) est un problème combinatoire NP-Difficile qui a été formulé, pour la première fois, par Koopmans et Beckmann (1957). Il peut être représenté de la manière suivante : étant donné un certain nombre d'unités et de sites (ou localisations), la résolution du QAP consiste à affecter chaque unité à un site afin de minimiser le coût total des flux qui opèrent entre chaque paire d'unités. Ce coût dépend de la distance qui sépare les unités et du coût unitaire du flux entre celles-ci.

Dans la variante multi-objectifs proposée par Knowles et Corne (2003), nommée **MQAP** (*Multi-objective QAP*), différents types de flux sont pris en compte dans le problème. Pour résoudre le MQAP, il faut trouver la répartition des unités sur les différents sites qui permet de minimiser le coût total de chaque flux. Dans ce contexte, chaque type de flux correspond à

une fonction objectif du problème. Un exemple de MQAP avec 4 sites/unités est donné à la Figure 4.1.

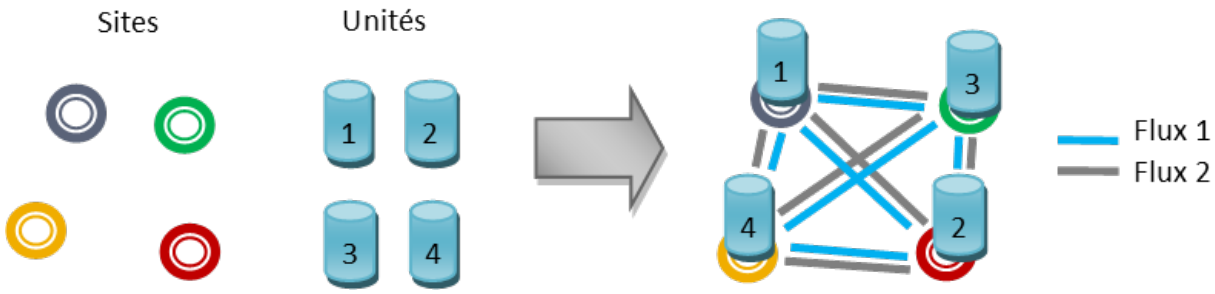


Figure 4.1: Exemple de problème d'assignation quadratique multi-objectifs

Plus formellement, la Définition 4 expose la formulation mathématique du MQAP à M objectifs.

Définition 4 *(Problème d'affectation quadratique multi-objectifs)*

$$\begin{aligned} &\textit{minimiser} \quad C(\pi) = \{C^1(\pi), C^2(\pi), ..., C^M(\pi)\} \\ &\textit{avec} \quad C^k(\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} b^k_{\pi_i \pi_j}, \quad k = 1, ..., M \end{aligned} \tag{4.1}$$

où n est le nombre de sites/unités, a_{ij} est la distance entre les sites i et j, $b^k_{\pi_i \pi_j}$ est le coût du k-ième flux entre l'unité π_i et l'unité π_j. Enfin, π_i correspond au i-ème élément dans la permutation π.

Instances de MQAP Knowles et Corne (2003) ont été les premiers à s'intéresser au problème de MQAP en proposant deux générateurs d'instances. Plusieurs paramètres sont gérés par ces générateurs : le nombre de sites/unités, le nombre d'objectifs (types de flux), les corrélations entre les objectifs, etc. Ces corrélations influent en grande partie sur la structure de l'espace de recherche et la forme du front Pareto. De plus, plusieurs instances de test, créées avec leurs générateurs et contenant de 10 à 30 variables de décision, sont proposées et peuvent être utilisées pour valider la qualité d'une méthode de résolution. Le Tableau 4.1 expose les caractéristiques des 22 instances de test : le nom de l'instance (*Nom*), le nombre de sites/unités (*Taille*), le type de générateur utilisé pour la création (uniforme ou réel), le nombre d'objectifs (*Obj.*) et le paramètre de corrélation entre le premier et les autres objectifs (Corr.).

Tableau 4.1: Instances de MQAP de 10 à 30 variables de décision générées par Knowles et Corne (2003)

Nom	Taille	Type	Obj.	Corr.
KC10-2fl-1uni	10	Uni.	2	0.0
KC10-2fl-2uni	10	Uni.	2	0.8
KC10-2fl-3uni	10	Uni.	2	-0.8
KC20-2fl-1uni	20	Uni.	2	0.0
KC20-2fl-2uni	20	Uni.	2	0.7
KC20-2fl-3uni	20	Uni.	2	-0.7
KC30-3fl-1uni	30	Uni.	3	0.0
KC30-3fl-2uni	30	Uni.	3	0.4
KC30-3fl-3uni	30	Uni.	3	-0.4
KC10-2fl-1rl	10	Réel	2	0.0
KC10-2fl-2rl	10	Réel	2	0.7
KC10-2fl-3rl	10	Réel	2	-0.7
KC10-2fl-4rl	10	Réel	2	0.7
KC10-2fl-5rl	10	Réel	2	0.7
KC20-2fl-1rl	20	Réel	2	0.0
KC20-2fl-2rl	20	Réel	2	0.4
KC20-2fl-3rl	20	Réel	2	-0.4
KC20-2fl-4rl	20	Réel	2	0.4
KC20-2fl-5rl	20	Réel	2	0.4
KC30-3fl-1rl	30	Réel	3	0.4
KC30-3fl-2rl	30	Réel	3	0.7
KC30-3fl-3rl	30	Réel	3	-0.4

Tableau 4.2: Instances de MQAP de 60 variables de décision générées par Garrett et Dasgupta (2009)

Nom	Taille	Type	Obj	Corr.
Gar60-2fl-1uni	60	Uni.	2	-0.3
Gar60-2fl-2uni	60	Uni.	2	0.0
Gar60-2fl-3uni	60	Uni.	2	0.3
Gar60-2fl-4uni	60	Uni.	2	-0.8
Gar60-2fl-5uni	60	Uni.	2	0.8
Gar60-2fl-1rl	60	Réel	2	-0.3
Gar60-2fl-2rl	60	Réel	2	0.0
Gar60-2fl-3rl	60	Réel	2	0.3
Gar60-2fl-4rl	60	Réel	2	-0.8
Gar60-2fl-5rl	60	Réel	2	0.8
Gar60-3fl-1uni	60	Uni.	3	0.0
Gar60-3fl-2uni	60	Uni.	3	-0.5
Gar60-3fl-3uni	60	Uni.	3	0.5
Gar60-3fl-1rl	60	Réel	3	0.0
Gar60-3fl-2rl	60	Réel	3	-0.5
Gar60-3fl-3rl	60	Réel	3	0.5
Gar60-4fl-1uni	60	Uni.	4	0.0
Gar60-4fl-2uni	60	Uni.	4	-0.5
Gar60-4fl-3uni	60	Uni.	4	0.5
Gar60-4fl-1rl	60	Réel	4	0.0
Gar60-4fl-2rl	60	Réel	4	-0.5
Gar60-4fl-3rl	60	Réel	4	0.5

Garrett et Dasgupta (2009) ont utilisé ce générateur pour créer un ensemble d'instances de 60 variables de décision. Les noms des instances et leurs caractéristiques sont contenus dans le Tableau 4.2. Finalement, Paquete et Stützle (2006) ont également produit des instances de test de 25, 50 et 75 unités en utilisant des paramètres de corrélations différents pour chaque configuration.

Résolution du MQAP dans la littérature Plusieurs travaux de la littérature s'intéressent à l'optimisation du MQAP et sont présentés ci-après. Dans leurs études empiriques respectives, Garrett et Dasgupta (2009) et López-Ibáñez *et al.* (2006) analysent les performances de l'hybridation d'un algorithme évolutionnaire avec une forme de recherche locale. Drugan et Thierens (2012) introduisent, quant à eux, un algorithme de recherche locale Pareto (PLS) avec

un mécanisme de perturbation stochastique pour sortir des optimums locaux. Un algorithme mémétique parallèle asynchrone, nommé PasMoQAP, a également été proposé par Sanhueza *et al.* (2017) pour répondre au problème de MQAP. Ce dernier reprend le principe de modèle en îles avec des migrations de solutions à intervalles réguliers. Outre les algorithmes mémétiques, d'autres types d'approches ont été étudiés. Li et Landa-Silva (2009) proposent ainsi un algorithme glouton élitiste et stochastique avec, encore une fois, une forme de recherche locale. Pour résoudre le MQAP, Özkale et Fığlalı (2013) adaptent plusieurs algorithmes de colonie de fourmis multi-objectifs. Enfin, plus récemment, Samanta *et al.* (2018) modifient un algorithme de colonie d'abeilles pour répondre à une variante du MQAP à deux objectifs : le bi-d-QAP dans lequel une fonction objectif correspond à un QAP uni-objectif.

4.1.1.2 LE PROBLÈME DU VOYAGEUR DE COMMERCE MULTI-OBJECTIFS

Formulation du problème Le problème du voyageur de commerce, ou **TSP** (*Travelling Salesman Problem*), est un problème NP-Difficile largement étudié dans le domaine de l'optimisation combinatoire puisque ses applications réelles sont nombreuses et variées dans les domaines de l'ingénierie ou de la génétique par exemple (Punnen, 2007). De plus, grâce notamment à sa simplicité, les instances de TSP ont largement été utilisées pour valider des approches ou des méthodes de résolution dans la littérature.

À partir d'un ensemble de villes séparées par des distances définies, la résolution du TSP consiste à déterminer le plus court chemin qui passe par toutes les villes une fois et qui retourne à la ville de départ. Afin d'illustrer le problème, un exemple de TSP est donné à la Figure 4.2. Plus formellement, soit un graphe pondéré $G = (V, E)$ avec V un ensemble de sommets (villes) et E les arêtes (chemins) reliant les sommets. Le but du TSP est de trouver le cycle hamiltonien de poids minimum dans le graphe G. Un cycle hamiltonien étant un cycle qui passe une et une seule fois par chaque sommet.

Dans la variante multi-objectifs du TSP, le MOTSP (Multi-Objective TSP) (Borges et Hansen, 2002), plusieurs types de coûts sont définis entre chaque paire de villes et sont associés à des matrices de coût particulières. Ces coûts peuvent correspondre à des durées de trajet, des frais de trajets (e.g. essence consommée, prix des péages) ou tout simplement à des distances comme dans la version uni-objectif. La formulation mathématique du MOTSP à M objectifs est présentée à la Définition 5.

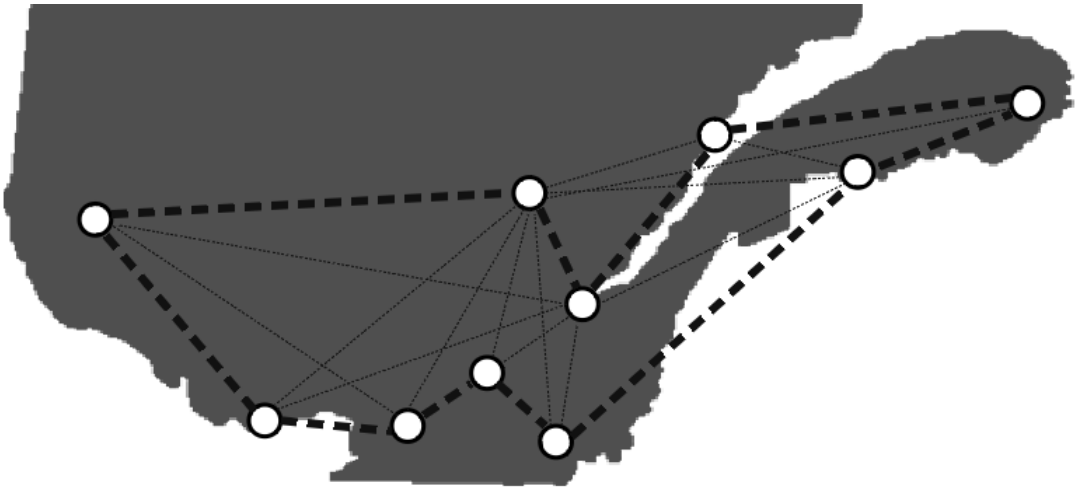


Figure 4.2: Exemple de problème du voyageur de commerce

Définition 5 *(Problème du voyageur de commerce multi-objectifs)*

$$\begin{aligned} &min \quad F(\pi) = ((f_1(\pi), f_2(\pi), ..., f_M(\pi))) \\ &avec \quad f_k(\pi) = \sum_{i=1}^{n-1} d_{\pi_i,\pi_{i+1}}(k) + d_{\pi_n,\pi_1}(k) \\ &k = 1, ..., M \end{aligned} \tag{4.2}$$

avec n le nombre de villes, $\pi = (\pi_1, \pi_2, ..., \pi_n)$ une solution sous forme de permutation de villes et $d_{i,j}(k)$ le coût entre la ville i et la ville j pour le k-ième objectif.

Instances de MOTSP symétriques Dans le cadre de cette recherche, des instances de MOTSP symétriques [1] sont résolues. Pour ce faire, la bibliothèque TSPLIB (Reinelt, 1990) propose des instances de test pour le problème uni-objectif du TSP : kroAB100, kroB100, etc. Ces dernières ont souvent été combinées pour former des instances de MOTSP (Jaszkiewicz, 2002) (Ke *et al.*, 2013). Par exemple, l'instance kroAB200 est une instance bi-objectif avec 200 villes et représente la combinaison de kroA200 (1er objectif) et kroB200 (2ème objectif). Ces instances présentent des fronts Pareto fournis en nombre de solutions, convexes et continus (Paquete et Stützle, 2003).

Dans leur travail, Paquete et Stützle (2009) ont créé des instances de 100, 300 et 500

1. La distance d'une ville i à une ville j est la même que celle de j à i

villes. Ils utilisent un générateur aléatoire conçu initialement pour créer des instances de TSP (McGeoch, 2001). Pour chaque fonction objectif, ils calculent une matrice de distance grâce au générateur. Trois classes d'instances sont considérées : *Euclidean*, *Random* et *Mixed*. Dans la première, les villes sont placées aléatoirement dans un espace à deux dimensions et les distances euclidiennes entre les villes sont tout simplement calculées afin de générer les matrices de distance. Pour les instances *Random*, les distances entre les villes sont déterminées aléatoirement dans un intervalle défini. Enfin, comme son nom l'indique, la classe *Mixed* combine les deux précédentes, de sorte que les matrices sont générées par chacune des deux méthodes. Une autre classe d'instances a également été proposée par Lust et Teghem (2010c) et est nommée *Clustered*. Cette fois-ci, les villes sont réparties par groupe et non pas totalement aléatoirement. Les distances euclidiennes entre les villes sont utilisées pour définir les matrices de distance.

Tout comme la version uni-objectif, le MOTSP est un des problèmes combinatoires multi-objectifs les plus étudiés dans la littérature (Lust et Teghem, 2010b). De multiples heuristiques ont été proposées pour traiter ce problème et sont présentées ci-après.

Résolution du MOTSP dans la littérature Des algorithmes basés sur la recherche locale Pareto (PLS, voir Section 1.5.3), ont été proposés et adaptés pour le MOTSP. Dans la méthode de Angel *et al.* (2004), l'idée est de déterminer toutes les solutions qui ne dominent pas la solution courante. Dans l'approche de Paquete *et al.* (2004), un PLS est également appliquée pour résoudre un MOTSP et plusieurs types de voisinage sont testés dont le 3-opt qui permet d'obtenir de meilleurs résultats, mais requiert un temps de calcul très élevé. Quelques années plus tard, Lust et Teghem (2010c) ont proposé l'algorithme 2PPLS qui permet en deux étapes d'optimiser un MOTSP. La première étape consiste à trouver les solutions supportées du problème, c'est-à-dire les solutions qui peuvent être trouvées par une méthode d'agrégation. Dans la seconde étape, une PLS est appliquée sur ces solutions afin de trouver les solutions non-supportées. Dans leur travail, une méthode est proposée afin de limiter le nombre de 2-opt effectués lors de la recherche locale.

À la manière de l'algorithme MOEA/D (Qingfu et Hui, 2007), des approches basées sur la décomposition ont été proposées. Jaszkiewicz (2002) utilise la méthode de Tchebycheff pour générer les sous-problèmes uni-objectif. De même, l'algorithme MOEA/D-ACO (Ke *et al.*, 2013) hybride un algorithme de colonie de fourmis avec l'algorithme MOEA/D. Chaque

colonie est alors responsable de la résolution d'un sous-problème uni-objectif. Enfin, dans l'algorithme MOMAD proposé par Ke *et al.* (2014), les grandes lignes de 2PPLS et MOEA/D sont combinées. Deux populations sont maintenues : une population pour garder les meilleures solutions de chaque sous-problème et une population contenant les solutions pour la PLS.

4.1.2 L'ALGORITHME ÉVOLUTIONNAIRE GISMOO

Les expérimentations des modèles parallèles sont basées sur l'algorithme GISMOO (Zinflou *et al.*, 2012) qui est relativement récent et qui s'est montré compétitif par rapport à des algorithmes tels que NSGA-II ou SPEA-II en termes de convergence vers le front Pareto et de diversité des solutions. De plus, à notre connaissance, aucun travail concernant sa parallélisation n'a été effectué. Avant de comparer différents modèles distribués, les performances de GISMOO sont évaluées par rapport à d'autres AEMO dans un contexte d'implémentation commun et des conditions expérimentales similaires. Pour ce faire, l'implémentation de GISMOO au sein d'une infrastructure logicielle, communément appelée *framework*, a été réalisée. Il existe plusieurs frameworks pour implémenter des métaheuristiques et certains d'entre eux sont décrits, après avoir rappelé les grandes lignes de GISMOO.

Le comportement de l'algorithme GISMOO (Zinflou *et al.*, 2012), qui a été sommairement présenté dans le premier chapitre, est maintenant détaillé. Le fonctionnement général de l'algorithme correspond à celui d'un AEMO basé sur la dominance Pareto, mais il incorpore une phase immune dans laquelle de nouveaux individus sont générés par opérations empruntées aux algorithmes de systèmes immunitaires artificiels. L'Algorithme 10 fournit un pseudo-code du comportement de GISMOO. Tout d'abord (lignes 1-5), les populations Parent POP et Enfant Q sont initialisées avec N individus et les fonctions objectif sont évaluées. L'archive de solutions non-dominées A est mise à jour en fonction de POP et Q. Ensuite, le processus itératif principal est répété jusqu'à atteindre un critère d'arrêt qui est généralement un maximum de générations ou un temps d'exécution maximum. Le processus peut être décomposé en trois phases : la phase de remplacement, la phase génétique et la phase immune. Dans la première, les populations POP et Q sont combinées et les facteurs de dominance et d'isolement sont affectés à chaque individu (ligne 6). La nouvelle population $POP \cup Q$ est triée selon ces deux facteurs en priorisant le facteur de dominance (ligne 7). Ce tri permet d'effectuer le remplacement élitiste (ligne 8) en conservant les N meilleurs individus de $POP \cup Q$.

Dans la phase génétique (lignes 9-15), $N/2$ individus enfants sont ajoutés à Q. Pour ce

faire, des opérations classiques aux AEMO sont effectuées : une sélection par tournoi binaire basé sur les facteurs, un croisement et une mutation probabiliste. Néanmoins, il faut noter que pour chaque nouvel enfant, deux individus potentiels sont créés, mais seul le meilleur est ajouté à Q. Dans ce contexte, le meilleur individu correspond à celui qui domine l'autre, mais en cas d'indifférence entre les deux solutions, un individu est choisi aléatoirement.

Pour compléter la population Enfant Q, la phase immune (lignes 16-26) génère également $N/2$ individus. Tout d'abord, la population doit être triée par front à la manière de NSGA-II (Deb *et al.*, 2002). Pour chaque individu du premier front, un nombre *nb_clones* de clones est créé. Le calcul de *nb_clones* est défini à la ligne 19 de l'algorithme et prend en compte le facteur d'isolement *Dist* de l'individu. Chaque clone est modifié par une hypermutation (ligne 22), une forme de mutation dont l'ampleur de la modification dépend de la qualité de la solution, c'est-à-dire le facteur d'isolement de l'individu. Tout comme dans la phase génétique, deux clones sont créés et seul le meilleur est ajouté à Q. Les trois phases sont répétées jusqu'à atteindre la condition d'arrêt préalablement définie. Une fois le processus itératif terminé, l'archive de solutions non-dominées, qui a continuellement été mis à jour après les évaluations des nouvelles solutions, est retournée (ligne 28).

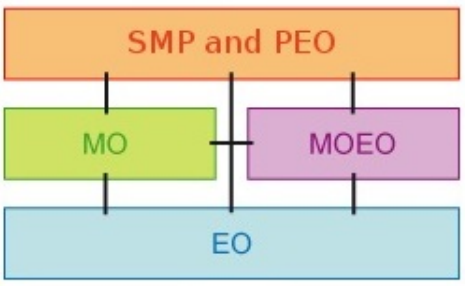
Algorithme 10: Pseudo-code de GISMOO (Zinflou *et al.*, 2012)

Initialiser l'archive $A = \varnothing$;
Initialiser aléatoirement la population Parent POP;
Initialiser la population Enfant Q par croisement des individus de POP;
Évaluer POP et Q et mettre à jour A;
Tant que *condition d'arrêt non atteinte* **Faire**
    `/* Remplacement */`
    Calculer les facteurs de la population $POP \cup Q$;
    Trier $POP \cup Q_t$;
    Conserver les N meilleurs individus de $POP \cup Q$ dans POP;
    `/* Phase génétique */`
    Tant que $|Q| < N/2$ **Faire**
        Sélectionner deux parents P_1 et $P_2 \in POP$;
        Croiser P_1 et P_2 pour créer deux enfants E_1 et E_2;
        Évaluer E_1 et E_2 et mettre à jour A;
        Muter de manière probabiliste E_1 et E_2;
        Ajouter le meilleur enfant à Q;
    FinTantque
    `/* Phase Immune */`
    Trier par front POP;
    Total = somme des facteurs d'isolement des individu du front 1;
    Pour *chaque individu x du premier front* **Faire**
        $nb\_clones = \lceil (Dist(x) * N/2)/Total \rceil$;
        Pour *i allant de 1 à nb_clones* **Faire**
            c_1 et c_2, clones de x;
            Hypermuter c_1 et c_2;
            Évaluer c_1 et c_2 et mettre à jour A;
            Ajouter le meilleur clone à Q;
        FinPour
    FinPour
FinTantque
retourner A ;

4.1.3 LE FRAMEWORK PARADISEO

Implémenter un algorithme de résolution multi-objectifs en totalité est souvent long et fastidieux. Pour faciliter et accélérer ce processus, plusieurs frameworks et bibliothèques *open source* ont été proposés. Ils permettent d'utiliser du code déjà implémenté par de tierces personnes et de réutiliser son propre code par la suite. De plus, un framework permet de comparer, de manière équitable, des métaheuristiques par des métriques déjà implémentées. En effet, le code utilisé pour les différents algorithmes de résolution suit une certaine norme et l'environnement d'expérimentation peut être exactement le même pour tous. Quelques outils ont été proposés pour implémenter des métaheuristiques multi-objectifs : la plateforme de programmation **PISA** (Bleuler *et al.*, 2003), le framework orienté objet **Open BEAGLE** (Gagné et Parizeau, 2006) ou encore l'infrastructure logicielle **jMetal** (Durillo et Nebro, 2011).

Dans cette thèse, le framework **ParadisEO** (Cahon *et al.*, 2004), qui permet la réalisation d'algorithmes inspirés de métaheuristiques, est utilisé. Il est composé de quatre modules (Figure 4.3) : les modules *MO* et *EO* pour créer des métaheuristiques basées sur une population ou sur un individu, le module *MOEO* qui facilite l'implémentation d'algorithmes multi-objectifs et le module *SPM and PEO* pour paralléliser les métaheuristiques à travers des modèles définis. Le code réalisé au sein de ParadisEO est réutilisable et peut être porté sur plusieurs systèmes d'exploitation. Comme ce framework propose certains algorithmes de référence comme NSGA-II ou SPEA-II et des outils pour implémenter un AEMO, l'utilisation de celui-ci est un choix approprié pour l'implémentation de GISMOO.



MO : Métaheuristiques basées sur un individu
EO : Métaheuristiques basées sur une population
MOEO : Optimisation multi-objectifs
SPM and PEO : Modèles hybrides et parallèles

Figure 4.3: Les quatre modules de Paradiseo (Cahon *et al.*, 2004)

4.1.4 IMPLÉMENTATION DE GISMOO DANS PARADISEO

Dans cette partie, l'implémentation dans Paradiseo de l'algorithme GISMOO, dont le comportement a été décrit à la Section 4.1.2, est détaillée. Afin d'être générique pour permettre l'implémentation facile et rapide de divers algorithmes, Paradiseo intègre les concepts de *templates* disponibles en `C++`. Ainsi, la plupart des classes sont implémentées comme des *templates* et permettent de prendre en compte n'importe quel type d'individu, de solution ou de problème. Les principales classes implémentées pour GISMOO sont décrites par la suite.

Algorithme général La structure générale de l'algorithme GISMOO est contenue dans la classe *moeoGISMOO* qui est dérivée de la classe *moeoEA* qui permet, comme son nom le suggère, l'implémentation de la structure d'un algorithme évolutionnaire multi-objectifs. L'Algorithme 11 présente les grandes lignes de la classe *moeoGismoo*. Un processus itératif est répété afin d'évoluer les populations sur plusieurs générations. Dans celui-ci, les phases génétique et immune sont appelées pour compléter la population Enfant à partir de la population Parent. Lorsque tous les nouveaux individus sont générés et évalués, l'opération de remplacement élitiste est réalisée et permet de conserver les meilleurs individus.

Algorithme 11: Algorithme de la classe *moeoGISMOO*

```
1 Initialisation des populations ;
2 Tant que condition d'arrêt non atteinte Faire
3 |   Générer des enfants par opérations génétiques ;
4 |   Générer des enfants par opérations immunes ;
5 |   Remplacement ;
6 FinTantque
7 retourner l'archive de solutions non-dominées ;
```

Pour qualifier les solutions et les différencier lors du tournoi binaire et la phase de remplacement, GISMOO attribue à chaque solution un facteur de dominance spécifique. Pour cette attribution, une nouvelle classe nommée *moeoGismooFitnessAssignment* a été implémentée.

Phase génétique La phase génétique de GISMOO est semblable à ce qui est déjà disponible sur ParadisEO. La sélection est effectuée via un tournoi binaire et la classe *moeoDetTournamentSelect* est réutilisée pour la simuler. Comme l'algorithme génère deux candidats potentiels pour chaque nouvel enfant, des adaptations sont nécessaires. Les opérations génétiques de

GISMOO sont appliquées au sein de la fonction *genOP* dans la nouvelle classe *moeoGismooGenOp*. Elle dérive de *eoBinOP*, qui génère un unique enfant à partir de deux parents présélectionnés. L'Algorithme 12 présente le pseudo-code de la fonction contenue dans *moeoGismooGenOp*. Tout d'abord, deux enfants sont créés par croisement des deux individus parents passés en paramètres. Une fois les nouveaux individus évalués, les solutions sont potentiellement ajoutées à l'archive. Une mutation probabiliste est ensuite effectuée avec une nouvelle mise à jour de l'archive. Finalement, l'individu enfant qui domine au sens Pareto l'autre est retourné par la fonction.

Algorithme 12: Algorithme de la classe *moeoGismooGenOp*

```
1  Fonction genOp(parent1,parent2)
2      enfant1 = croisement(parent1,parent2);
3      enfant2 = croisement(parent2,parent1);
4      évaluation(enfant1);
5      évaluation(enfant2);
6      archive(enfant1,enfant2);
7      si rand(0,1) < taux de mutation alors
8          mutation(enfant1);
9          mutation(enfant2);
10         évaluation(enfant1);
11         évaluation(enfant2);
12         archive(enfant1,enfant2);
13     finsi
14     Retourner le meilleur enfant;
```

Phase immune Comme les algorithmes de systèmes immunitaires artificiels ne sont pas implémentés dans ParadisEO, il est nécessaire de redéfinir de nouvelles classes pour la phase immune. Dans cette phase, il faut parcourir les solutions non-dominées selon le facteur de diversité et de calculer pour chaque solution un nombre de clones spécifique. Pour ce faire la classe *eoPopulator* de Paradiseo est réutilisée au sein de la nouvelle classe *moeoFrontPopulator*. Finalement, pour générer les nouveaux clones à partir d'un parent, la classe *moeoImmuneOp* a été créée et est présentée à l'Algorithme 13. Après l'hypermutation des deux clones, le meilleur clone, au sens Pareto, est retourné.

Problème MQAP L'algorithme GISMOO dans ParadisEO est générique et permet d'optimiser n'importe quel type de problème. Néanmoins, afin de résoudre un problème spécifique,

Algorithme 13: Algorithme de la classe *moeoImmuneOp*

```
1 Fonction GenererClone(parent)
2     clone1 = parent ;
3     clone2 = parent ;
4     hypermutation(clone1);
5     hypermutation(clone2);
6     Retourner le meilleur clone ;
```

il est nécessaire d'implémenter le problème sous ParadisEO. C'est-à-dire définir les fonctions objectif, la fonction d'évaluation et les variables de décision. Le génotype d'un individu pour le MQAP est décrit dans la classe *MQAPIndividual*. Il faut également définir les croisements et mutations désirés pour la résolution du problème. Pour le croisement, le *cycle crossover* a été choisi et est implémenté dans la nouvelle classe *eoCycleCrossover*. En ce qui concerne la mutation, la classe *eoInversionMutation* a été créée pour utiliser la mutation par inversion.

Résumé des classes implémentées La Figure 4.4 permet de résumer les principales classes implémentées dans ce travail et leur relation d'héritage avec les classes originales de ParadisEO. Les classes dépendantes de l'AEMO sont présentées dans la partie supérieure de la Figure alors que dans la partie inférieure les classes qui sont dépendantes du problème sont exposées. Cette synthèse des classes implémentées met en évidence la réutilisation de code effectué, en partie, grâce à la spécialisation de certaines classes disponibles dans ParadisEO.

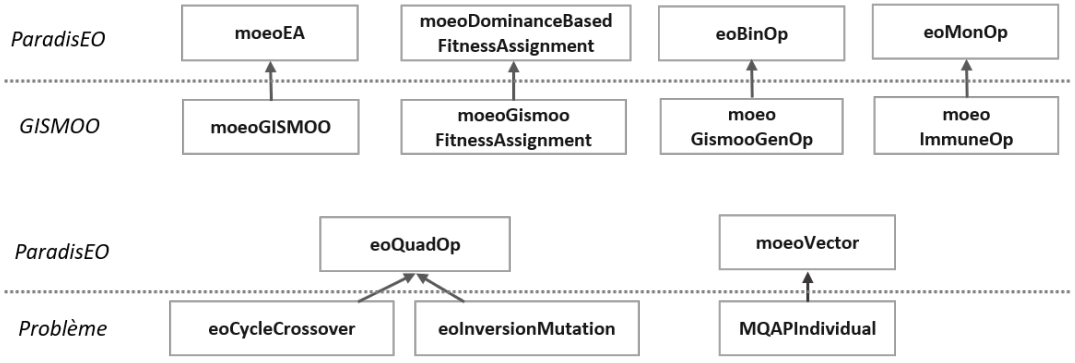


Figure 4.4: Vue d'ensemble des principales classes implémentées dans Paradiseo

4.1.5 EXPÉRIMENTATION DE GISMOO DANS PARADISEO

L'expérimentation est divisée en deux phases. La première phase permet de comparer l'algorithme GISMOO aux algorithmes de référence NSGA-II et SPEA-II dans le framework ParadisEO. Puis, dans la seconde phase, les temps d'exécution de l'implémentation de GISMOO dans ParadisEO sont comparés aux temps d'exécution obtenus avec une implémentation complète de l'algorithme.

Configuration de l'expérimentation Les différents algorithmes ont été exécutés sur un ordinateur standard possédant un processeur Intel®Core^{TM} i5-4200M (2,5 GHz) et une mémoire vive de 8 Go. En suivant les recommandations des auteurs (Zinflou *et al.*, 2012), toutes les expérimentations GISMOO ont été réalisées avec les paramètres suivants : taille de la population, probabilité de mutation et probabilité de croisement sont respectivement fixées à 100, 0.06 et 1.0.

Les problèmes de test pour le MQAP, proposés par Knowles et Corne (2003), sont résolus dans le cadre de cette expérimentation. Ces instances, dont les caractéristiques sont détaillées à la Section 4.1.1.1, représentent des problèmes à 2 ou 3 objectifs et impliquent 10, 20 ou 30 sites. Elles ont été résolues 10 fois pour chaque algorithme et ce sont les moyennes des différentes métriques qui ont été retenues. Dans un soucis de lisibilité, les instances sont groupées par type, c'est-à-dire le nombre de variables de décision et le modèle de l'instance réels (rl) ou uniformes (uni). Ainsi, les résultats du groupe *10 -uni* correspondent aux résultats moyens obtenus par les instances *10-2fl-1uni*, *10-2fl-2uni* et *10-2fl-3uni*. Il en est de même pour les autres groupes d'instances.

Pour la comparaison entre les différents AEMO, des temps d'exécution maximums sont fixés (35, 70 et 105 secondes) en fonction du nombre de variables de décision. Dans le cadre de cette expérimentation, trois métriques (Section 1.2) sont utilisées pour évaluer la qualité des ensembles de solutions obtenues : la métrique de convergence *GD*, la couverture de deux ensembles $\mathscr{C}$ et la métrique d'espacement minimal *ms*. Pour le calcul de la métrique *GD*, les fronts Pareto doivent être connus ou approximés. Les instances de test à seulement 10 sites ont des fronts Pareto qui sont donnés par les auteurs (Knowles et Corne, 2003). En revanche, pour les instances à 20 ou 30 sites, de bonnes approximations des fronts Pareto sont calculées et utilisées.

Comparaison de GISMOO à d'autres AEMO Dans cette première phase d'expérimentation, l'algorithme GISMOO est comparé aux algorithmes de référence NSGA-II et SPEA-II dans ParadisEO. Les Tableaux 4.3 et 4.4 exposent respectivement les valeurs moyennes obtenues pour les métriques *GD* et *ms* pour chaque AEMO.

Tableau 4.3: Convergence *GD* moyenne de GISMOO, NSGA-II et SPEA-II sur des instances de MQAP

	GISMOO	NSGA-II	SPEA-II
10 - uni	**94**	7480	7554
10 - rl	**6963**	151048	222637
20 - uni	**11797**	29133	30366
20 - rl	**692507**	2353885	2676517
30 - uni	**26641**	79503	118872
30 - rl	**1123041**	1679623	3324867

Tableau 4.4: Espacement minimal *ms* moyen de GISMOO, NSGA-II et SPEA-II sur des instances de MQAP

	GISMOO	NSGA-II	SPEA-II
10 - uni	**0,010**	0,016	0,133
10 - rl	**0,070**	0,108	0,142
20 - uni	**0,061**	0,090	0,124
20 - rl	**0,040**	0,066	0,112
30 - uni	**0,055**	0,102	0,102
30 - rl	**0,034**	0,101	0,088

D'après les résultats, GISMOO obtient toujours une plus faible valeur *GD*, ce qui indique qu'il trouve en moyenne des solutions plus proches des solutions Pareto optimales que les deux autres AEMO. À l'égard des résultats de la métrique de couverture $\mathscr{C}$ exposés aux Figures 4.5 et 4.6, ceux-ci sont, de nouveau, significativement à l'avantage de GISMOO. En effet, les valeurs de $\mathscr{C}$(NSGA-II,GISMOO) et de $\mathscr{C}$(SPEA-II,GISMOO) souvent proches de 0 indiquent que très peu de solutions trouvées par les algorithmes classiques dominent des solutions de GISMOO. Au contraire, les valeurs de $\mathscr{C}$(GISMOO,NSGA-II) et (GISMOO,SPEA-II) sont généralement grandes et montrent que GISMOO obtient une majorité de solutions qui dominent celles trouvées par NSGA-II et SPEA-II. Les métriques *GD* et $\mathscr{C}$ permettent de constater la capacité de convergence de GISMOO vers le front Pareto.

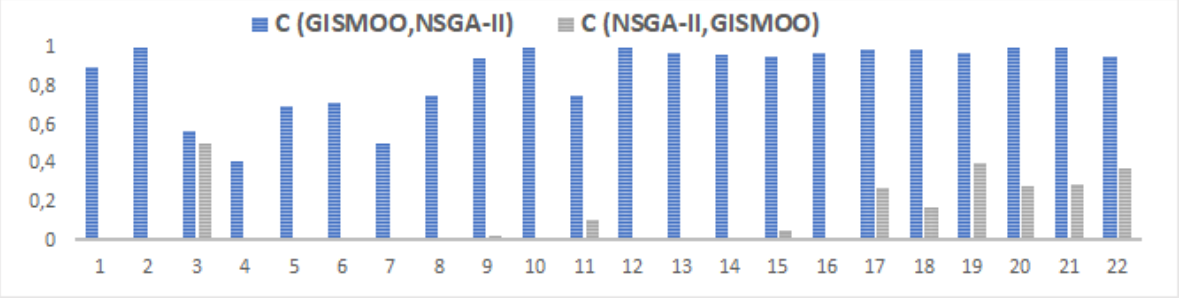


Figure 4.5: Couverture $\mathscr{C}$ moyenne entre GISMOO et NSGA-II sur 22 instances de MQAP

En ce qui concerne la diversité des solutions obtenues, les mesures d'espacement minimal

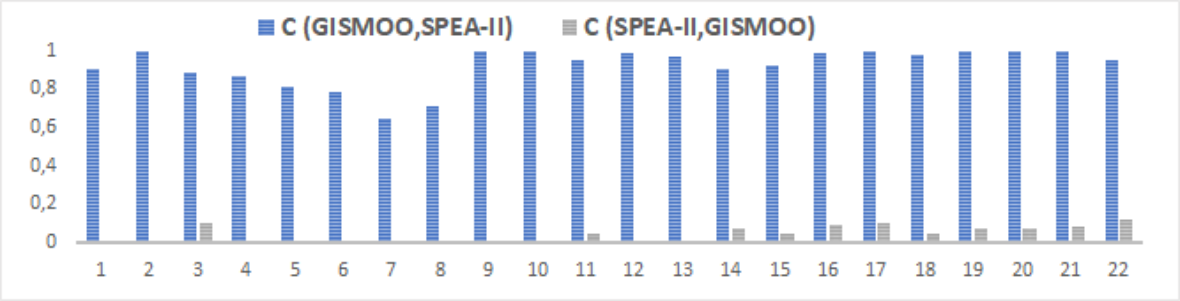


Figure 4.6: Couverture $\mathscr{C}$ moyenne entre GISMOO et SPEA-II sur 22 instances de MQAP

ms, présentées au Tableau 4.4, sont meilleures pour GISMOO pour toutes les instances testées. En effet, les valeurs sont, de manière significative, inférieures à celles de NSGA-II et SPEA-II. Ces résultats montrent l'efficacité de GISMOO à chercher des solutions dans des régions moins explorées et donc, à trouver un ensemble de solutions bien réparti sur le front Pareto. Cette habileté est, en partie, due à la phase immune de GISMOO qui permet d'intensifier l'exploration de l'espace de recherche.

Comparaison de deux implémentations de GISMOO Grâce à la flexibilité de ParadisEO, le comportement de la version originale de GISMOO peut être reproduit dans son intégralité. Ainsi, la qualité des solutions fournies par l'algorithme demeure la même. Cependant, l'utilisation d'un framework peut influencer sur le temps d'exécution de l'algorithme. La suite de cette expérimentation s'intéresse ainsi aux temps de résolution de deux implémentations de GISMOO, l'une réalisée avec Paradiseo (*Gismoo-Para*) et l'autre sans (*Gismoo-Seul*). La Figure 4.7 expose graphiquement les temps d'exécution moyens des deux versions pour chaque type d'instance et avec un nombre de générations fixe.

D'après les mesures obtenues, il apparaît que l'implémentation *Gismoo-Para* et jusqu'à deux fois plus lente (avec 10 variables) que la version *Gismoo-Seul*. Néanmoins, il faut noter que cette différence est moins importante lors de la résolution d'instances de 20 et 30 sites. Le type d'instance réel ou uniforme n'a pas d'influence sur le temps d'exécution, au contraire du nombre de variables de décision. Les évaluations des fonctions objectif sont particulièrement longues pour le MQAP ce qui explique, en grande partie, l'impact du nombre de variables sur le temps total d'exécution. Finalement, ces mesures suggèrent que la généricité offerte par la plateforme d'implémentation amène certains inconvénients comme un temps d'exécution plus grand.

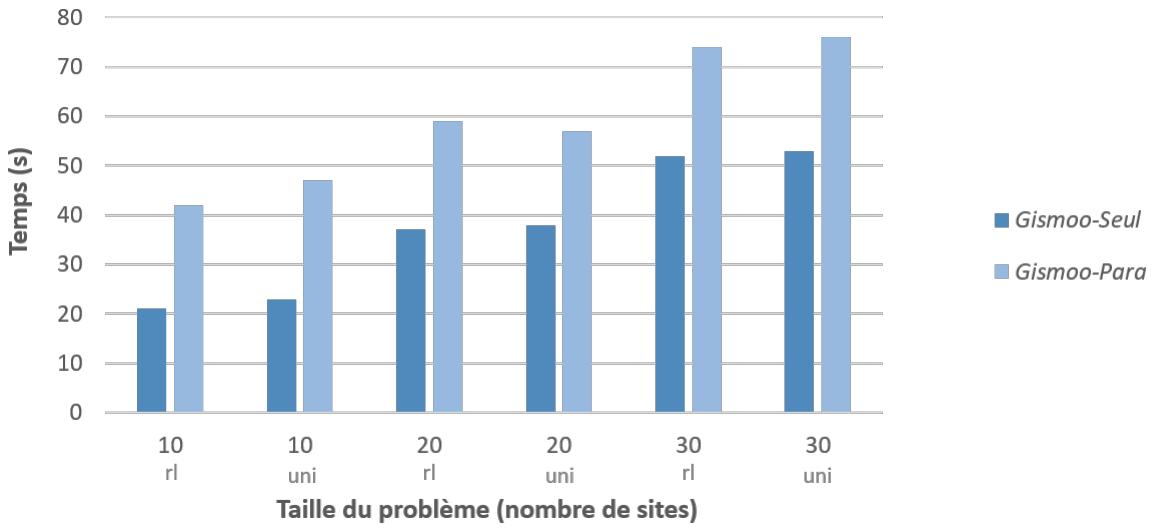


Figure 4.7: Temps d'exécution moyen en secondes pour les problèmes à 10, 20 et 30 sites

Conclusion sur l'implémentation dans ParadisEO L'implémentation de GISMOO dans le framework Paradiseo a permis de mettre en évidence l'efficacité de cet algorithme par rapport à des MOEA classiques dans des contextes d'implémentation et d'expérimentation communs. Son aptitude à trouver un ensemble de solutions diversifiées et proches du front Pareto a ainsi été exposée. Pour ces différentes raisons, l'utilisation de GISMOO comme algorithme de référence pour les recherches d'un modèle parallèle pour les AEMO est justifiée. Par ailleurs, même si l'utilisation d'un framework a certains avantages, elle ne permet pas nécessairement d'atteindre les meilleures performances en ce qui concerne les temps d'exécution. C'est la raison pour laquelle des implémentations autonomes seront privilégiées dans la suite des recherches au détriment de l'utilisation de framework ou de librairies.

4.2 COMPARAISON DES MODÈLES PARALLÈLES ET APM-MOEA

Les premières expérimentations visent principalement à montrer l'intérêt de la vue globale de l'île organisatrice et l'avantage des communications asynchrones dans le modèle APM-MOEA. En effet, cette étape constitue les prémices de l'analyse complète du modèle et permet de comparer son comportement vis-à-vis des principaux modèles en îles de la littérature sans l'ajout des îles témoins et sans les améliorations locales.

4.2.1 SÉLECTION DES MODÈLES COMPARÉS

Les modèles qui ont été sélectionnés pour la comparaison sont des modèles de la littérature qui ne sont pas uniquement applicables à un algorithme en particulier, mais qui peuvent s'adapter à une majorité d'AEMO. En outre, comme cette thèse s'intéresse principalement aux problèmes combinatoires, ils doivent permettre de les résoudre. À partir de ces critères, cinq modèles en îles ont été choisis parmi les modèles avec échange d'informations et les modèles qui découpent explicitement l'espace objectif (d'après la classification proposée à la Figure 2.3).

Dans la première catégorie, trois approches sont implémentées : un modèle qui utilise une topologie en anneau dans lequel les meilleurs individus sont échangés comme pMOMA (Fernández *et al.*, 2013), un dans lequel ce sont les premiers fronts obtenus par une assignation de rang qui sont partagés tel MFED (Essabri *et al.*, 2006) et enfin l'algorithme DRMOGA (Hiroyasu *et al.*, 2000) qui divise la population globale selon une fonction objectif et la distribue sur les différentes îles. Parmi les modèles qui divisent l'espace objectif, deux modèles sont considérés : le modèle de séparation en cônes (Branke *et al.*, 2004) qui sépare l'espace objectif de manière géométrique et le modèle de clustering (Streichert *et al.*, 2005) qui utilise des techniques de regroupement pour spécifier des zones à optimiser. Le Tableau 4.5 présente une synthèse de ces modèles, leur utilisation dans la littérature ainsi que les auteurs originaux.

Nom	Auteurs	Année	Problèmes
`pMOMA`	Fernández *et al.* (2013)	2013	combinatoires
`MFED`	Essabri *et al.* (2006)	2006	continus (ZDT)
`DRMOGA`	Hiroyasu *et al.* (2000)	2000	continus
`Séparation en cônes`	Branke *et al.* (2004)	2004	continus (ZDT)
`Clustering original`	Streichert *et al.* (2005)	2005	continus

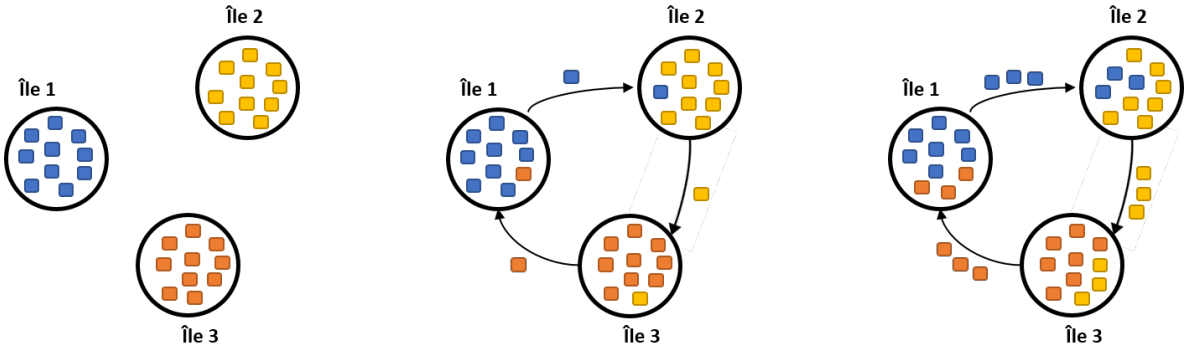
Tableau 4.5: Résumé des modèles en îles adaptés à GISMOO

4.2.2 ADAPTATION DES MODÈLES PARALLÈLES À GISMOO

En plus des cinq modèles présentés dans la section précédente, un modèle distribué sans échange d'information `IlesInde` a également été implémenté et sert de référence pour

l'ensemble des expérimentations. Sur chaque île de ce modèle, l'algorithme GISMOO fait évoluer une sous-population indépendamment des autres îles, tel qu'illustré à la Figure 4.8a. Dans chacun des modèles, l'archive finale contenant l'ensemble des solutions non-dominées est formée à partir des archives locales de chaque île.

Pour le modèle `pMOMA`, quelques solutions de chaque population sont échangées avec une population adjacente selon une topologie en anneau (Figure 4.8b). Les facteurs de dominance et d'isolement de GISMOO sont utilisés pour déterminer ces solutions et ainsi favoriser les solutions non-dominées les plus isolées. Dans le cadre de l'adaptation du modèle `MFED`, une assignation de rang similaire à NSGA-II (Deb *et al.*, 2002) a été implémentée afin de déterminer les fronts à échanger. Comme le suggère la Figure 4.8c, les échanges impliquent un nombre de solutions plus important que le modèle `pMOMA`. L'intégration de nouvelles solutions aux différentes populations est réalisée à l'aide d'un tri basé sur les facteurs de dominance et d'isolement de GISMOO.



(a) `IlesInde` **(b)** `pMOMA` **(c)** `MFED`

Figure 4.8: Représentation des modèles `IlesInde`, `pMOMA` et `MFED`

L'adaptation du modèle `DRMOGA` a rendu nécessaire la désignation d'une île organisatrice chargée de regrouper l'ensemble des solutions, de le trier selon une fonction objectif puis de répartir les solutions. La fonction objectif qui sert de base pour le tri change à chaque processus d'échange. Une représentation du modèle `DRMOGA` avec trois îles, dans lequel l'île 1 est l'organisatrice, est donnée à la Figure 4.10. Dans le cas des modèles de séparation de l'espace objectif `Séparation en cônes` et `Clustering original`, les principes de dominance sous contraintes (Deb *et al.*, 2002) ont été utilisés pour restreindre les processeurs à leur région spécifique. Ainsi, la fonction d'évaluation a été modifiée de sorte à rendre non réalisables les solutions n'appartenant pas à la zone à optimiser du processeur. La fonction de

dominance a également été adaptée dans le but de favoriser les solutions réalisables.

Afin de former les différentes régions pour le modèle `Séparation en cônes`, l'angle formé par le point Nadir et les solutions non-dominées extrêmes est subdivisé en k parties égales, où k correspond au nombre de processeurs disponibles. Un exemple de modèle en cônes à trois îles est présenté à la Figure 4.9 avec le point Nadir utilisé comme point de référence. Le modèle `Clustering original` utilise le partitionnement en k-moyennes sur l'archive de solutions non-dominées de GISMOO afin de déterminer les *centroïdes* qui représentent les régions à optimiser. La Figure 4.10 illustre le fonctionnement de ce modèle ainsi que le découpage effectué. Le comportement global est similaire à celui de `DRMOGA` en ce qui concerne le regroupement et la distribution des sous-populations, alors que la méthode de tri est différente. Finalement, pour le nouveau modèle `APM-MOEA`, les archives annexes contenant les nouvelles solutions non-dominées ont dû être créées et maintenues sur chaque île. Cependant, comme les solutions de l'archive utilisée dans GISMOO ne participent pas aux opérations évolutionnaires, le maintien d'une archive locale par île n'a pas était nécessaire. En effet, toutes les solutions non-dominées sont envoyées à l'île organisatrice pour mettre à jour l'archive globale.

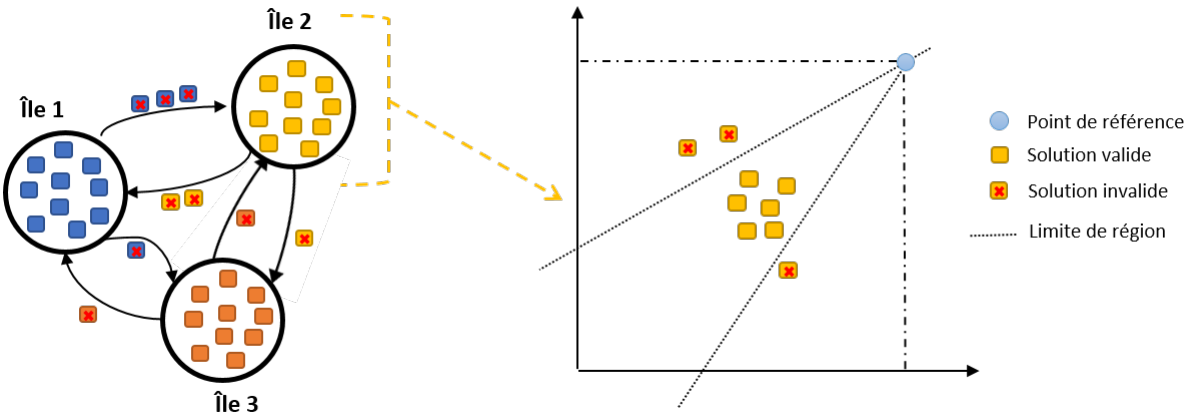


Figure 4.9: Représentation du modèle `Séparation en cônes`

4.2.3 CONDITIONS DE L'EXPÉRIMENTATION

L'expérimentation a été effectuée sur le supercalculateur ROMEO (ROMEO, 2018) basé au centre de calcul de la Maison de la Simulation de Champagne-Ardenne. Il dispose de 130 nœuds de calcul composés, entre autres, de 2 processeurs Intel Ivy Bridge (8 cœurs et 2,6

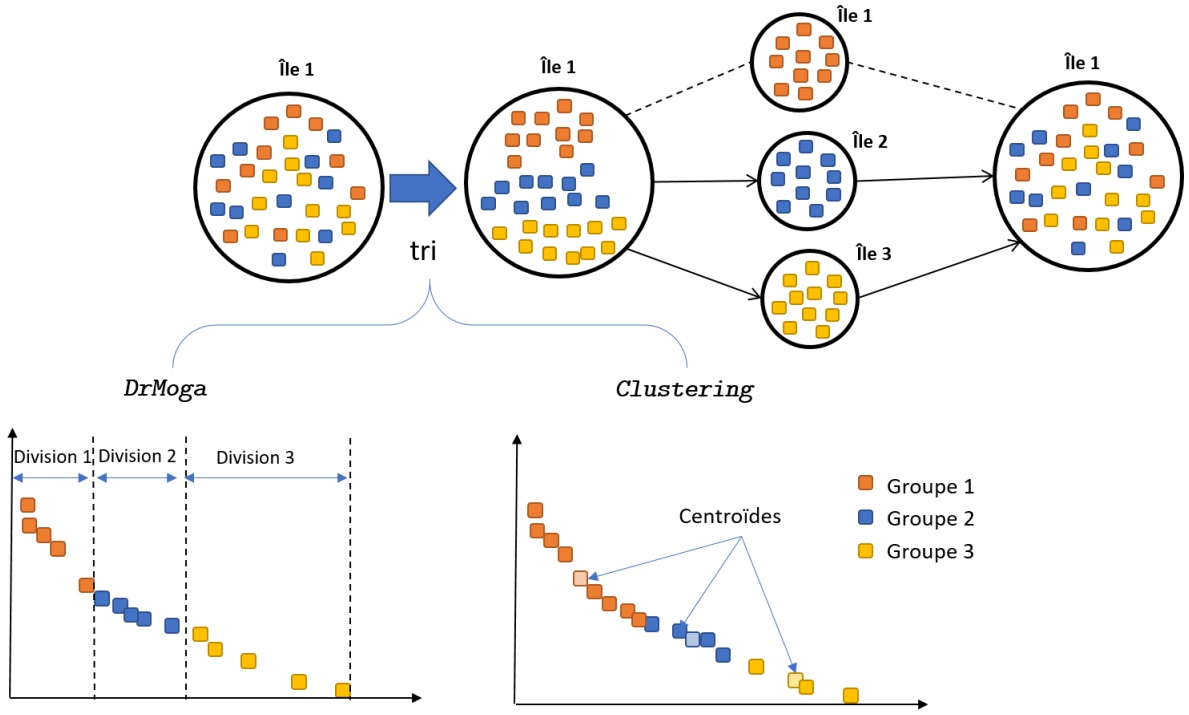


Figure 4.10: Représentation des modèles `DRMOGA` et `Clustering original`

GHz). Dans le but d'évaluer l'impact du nombre de processeurs utilisés, les implémentations sont testées avec 2, 4, 8 et 16 cœurs. De même, pour analyser l'influence du problème sur la performance des modèles parallèles, deux problèmes combinatoires bi-objectifs sont utilisés : le problème de voyageur de commerce multi-objectifs et le problème d'affectation quadratique multi-objectifs. Pour le MOTSP, les problèmes de test exposés à la Section 4.1.1.2 sont résolus et ceux-ci présentent des fronts Pareto convexes, continus et très fournis en solutions (Paquete et Stützle, 2003). En ce qui concerne le MQAP, le générateur d'instance précédemment introduit a été utilisé pour générer des instances de 30 à 500 sites qui ont des fronts généralement pauvres en solutions.

Le paramétrage de GISMOO est le même que celui recommandé par les auteurs et le nombre de générations maximum gen_{max} est défini par la formule suivante : $gen_{max} = 1500 + 10 * n_{var}$ où n_{var} correspond au nombre de variables de décision de l'instance. Les paramètres des modèles ont été choisis empiriquement. Ainsi, l'intervalle de migration pour chacun des modèles est fixé à 1 seconde et le nombre de solutions échangées pour le modèle `pMOMA` est de 10. Pour les opérations évolutionnaires, la mutation par inversion (Larrañaga *et al.*, 1999) et le croisement OX (Davis, 1985) sont utilisés.

Pour évaluer la qualité des solutions obtenues par les différents modèles, quatre métriques sont utilisées : la métrique de convergence *IGD*, l'hypervolume $\mathscr{S}$, la métrique de couverture de deux ensembles $\mathscr{C}$, et la mesure d'espacement minimal *ms*. Les approximations de fronts Pareto fournies par (Lust et Teghem, 2010c) sont utilisées comme ensemble de référence pour les deux premières métriques avec le problème de MOTSP. Pour le MQAP, les meilleures solutions trouvées par l'ensemble des modèles permettent de définir un front Pareto approximé. Toutes les instances de test ont été résolues 10 fois pour chaque modèle parallèle et les moyennes des valeurs des métriques sont présentées.

4.2.4 RÉSULTATS EXPÉRIMENTAUX DE LA COMPARAISON DES MODÈLES EN ÎLES

La convergence *IGD* correspond à la distance euclidienne moyenne des solutions du front Pareto approximé par rapport aux solutions de l'ensemble à évaluer. Pour calculer les valeurs de convergence *IGD* moyennes, les fonctions objectif ont été normalisées et les résultats sont présentés dans le Tableau 4.6. Pour chaque configuration selon le nombre d'îles, les meilleurs résultats sont indiqués en gras.

	MOTSP				MQAP			
	2 îles	4 îles	8 îles	16 îles	2 îles	4 îles	8 îles	16 îles
`IlesInde`	0,104	0,092	0,085	0,080	0,145	0,138	0,136	0,136
`pMOMA`	0,091	0,069	0,060	0,051	0,129	0,106	0,086	0,073
`DRMOGA`	0,127	0,092	0,084	0,088	0,142	0,123	0,096	0,070
`MFED`	0,113	0,105	0,096	0,097	0,136	0,108	0,094	0,074
`Séparation en cônes`	**0,051**	0,045	0,460	0,044	0,130	0,122	0,121	0,126
`Clustering original`	0,064	0,050	0,043	0,036	0,125	0,107	0,083	0,086
`APM-MOEA`	0,052	**0,033**	**0,025**	**0,018**	**0,123**	**0,099**	**0,076**	**0,057**

Tableau 4.6: Convergence *IGD* moyenne des sept modèles en îles sur les instances de MOTSP et de MQAP

En ce qui concerne la résolution du MOTSP, le modèle `Séparation en cônes` obtient en moyenne les meilleures valeurs de convergence *IGD* avec 2 îles. Il convient de noter que le modèle `APM-MOEA` fournit des solutions similaires en termes de convergence à celles du `Séparation en cônes` puisque les valeurs de convergence *IGD* sont relativement proches

(0,051 et 0,052). Avec l'utilisation d'un nombre plus important d'îles, `APM-MOEA` surclasse les modèles comparés et plus particulièrement avec 16 îles où il obtient une convergence *IGD* de 0,018 alors que le deuxième meilleur (`Clustering original`) obtient seulement 0,036. Pour certains modèles provenant de la littérature, les valeurs de convergence *IGD* ne diminuent que très légèrement avec l'augmentation du nombre d'îles ce qui montre leur difficulté à améliorer la qualité de solutions dans les plus grandes configurations. Les modèles de séparation de l'espace objectif, `Clustering original` et `DRMOGA`, fournissent en majorité les meilleurs résultats derrière `APM-MOEA`.

Pour les instances de MQAP, le modèle `APM-MOEA` démontre une capacité à mieux converger sur l'ensemble des configurations. Les plus grandes différences sont atteintes lors de l'utilisation de 16 îles ce qui suggère une meilleure extensibilité de `APM-MOEA` par rapport aux modèles comparés. Pour chacun des deux problèmes multi-objectifs et d'après la métrique de convergence *IGD*, le modèle `APM-MOEA` surclasse, en termes de convergence, les autres modèles dans la plupart des configurations et de manière plus significative avec un grand nombre d'îles.

Afin de confirmer les performances de `APM-MOEA` dans la plus grande configuration selon le nombre d'îles, la métrique de couverture $\mathscr{C}$ de deux ensembles a été calculée pour les modèles à 16 îles. La valeur de couverture $\mathscr{C}(A,B)$ indique le pourcentage de solutions obtenues par B qui sont dominées par au moins une solution de A. Les meilleurs modèles (selon la métrique de convergence *IGD*) ont été comparés pour chaque problème. Ainsi, pour le MOTSP, les ensembles de solutions fournis par le modèle `APM-MOEA` ont été directement comparés avec celles de `Séparation en cônes` et de `Clustering original` et les résultats sont présentés aux Figures 4.11 et 4.12. Pour le MQAP, les modèles `DRMOGA` et `Clustering original` sont ajoutés à la comparaison et les Figures 4.13 et 4.14 résument les résultats obtenus.

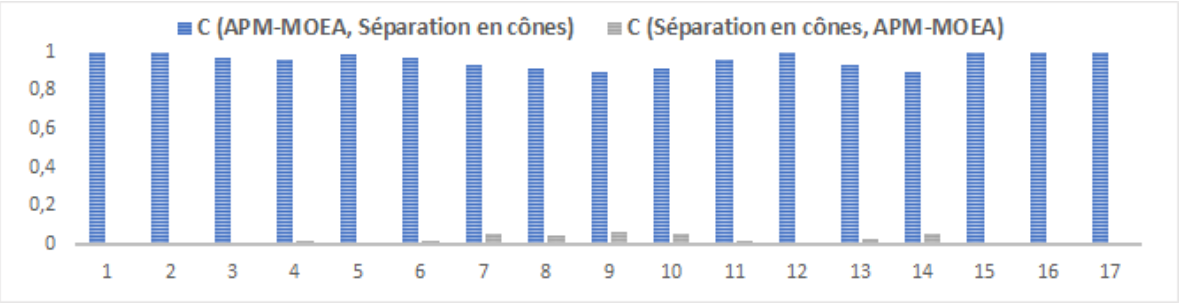


Figure 4.11: Couverture $\mathscr{C}$ moyenne entre `APM-MOEA` et `Séparation en cônes` pour les instances de MOTSP

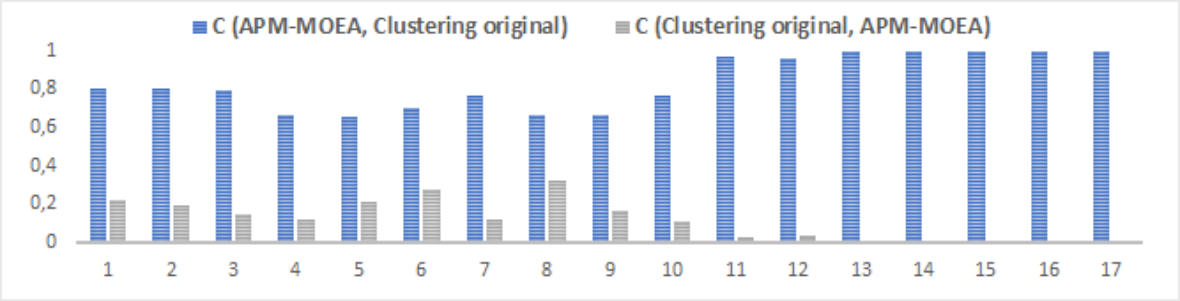


Figure 4.12: Couverture $\mathscr{C}$ moyenne entre `APM-MOEA` et `Clustering original` pour les instances de MOTSP

Tout d'abord, selon la Figure 4.11, la moyenne de $\mathscr{C}$(`APM-MOEA`,`Séparation en cônes`) varie entre $0,90$ et 1.00 ce qui indique que la grande majorité des solutions obtenues par le modèle `Séparation en cônes` sont dominées par les solutions fournies par `APM-MOEA`. Les valeurs de $\mathscr{C}$(`Séparation en cônes`, `APM-MOEA`) sont, au contraire, proches de 0 et prouvent que le modèle `APM-MOEA` trouve des solutions qui ne sont pas dominées par `Séparation en cônes`. À la Figure 4.12, la comparaison avec le modèle `Clustering original` montre des résultats similaires, même si ce dernier semble fournir quelques solutions de meilleure qualité. Concernant la résolution du MQAP, les valeurs de couverture $\mathscr{C}$, qui sont exposées aux Figures 4.13 et 4.14, montrent encore une fois la capacité de `APM-MOEA` à mieux converger vers le front Pareto que les modèles `DRMOGA` et `Clustering original`.

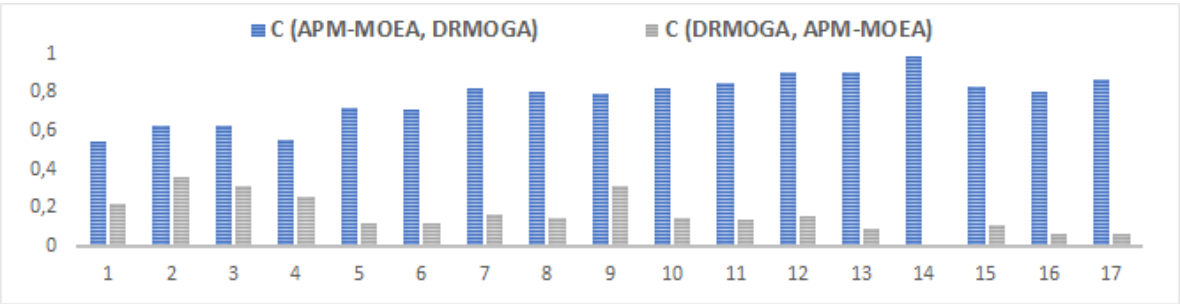


Figure 4.13: Couverture $\mathscr{C}$ moyenne entre `APM-MOEA` et `DRMOGA` pour les instances de MQAP

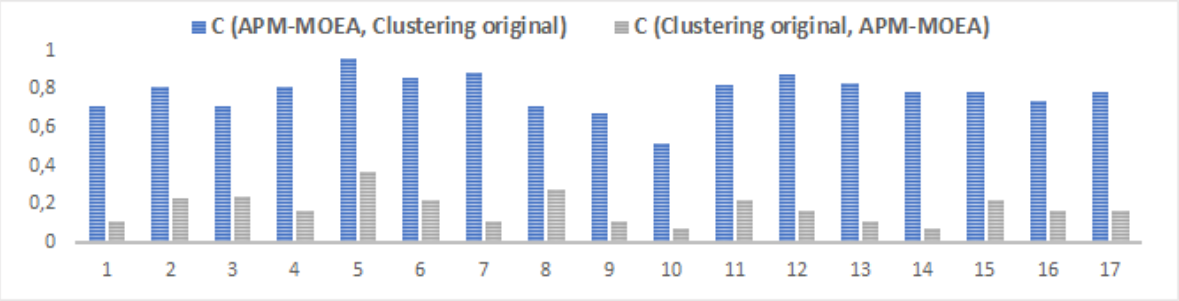


Figure 4.14: Couverture $\mathscr{C}$ moyenne entre `APM-MOEA` et `Clustering original` pour les instances de MQAP

Les deux premières métriques présentées permettent d'attester de la bonne convergence de `APM-MOEA`, mais ne permettent pas de vérifier la distribution de ses solutions. Pour répondre à cela, la métrique d'hypervolume $\mathscr{S}$ mesure la portion de l'espace objectif qui est dominée par l'ensemble de solutions considéré. De plus grandes valeurs d'hypervolume $\mathscr{S}$ sont préférables puisqu'elles correspondent à une meilleure couverture de l'espace objectif. Les valeurs d'hypervolume $\mathscr{S}$ moyennes pour tous les modèles comparés sont présentées dans les Figures 4.15 et 4.16 pour respectivement les instances de MOTSP et de MQAP.

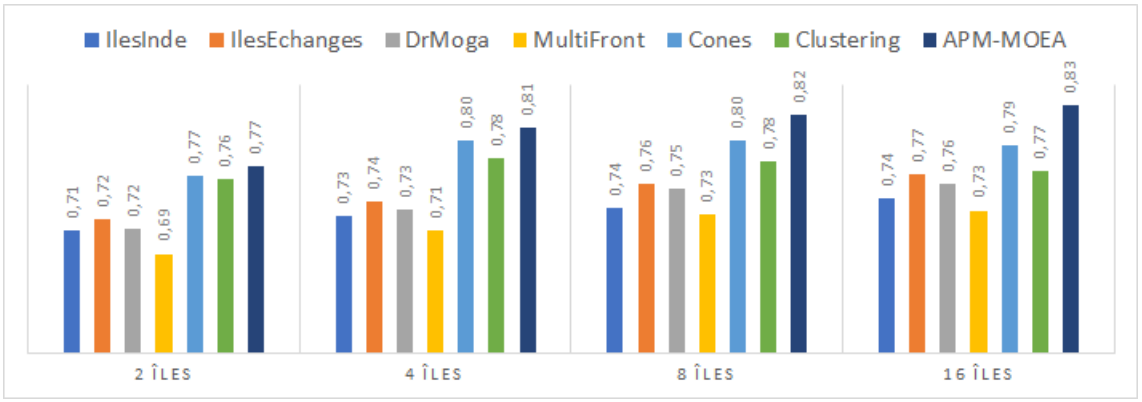


Figure 4.15: Hypervolume $\mathscr{S}$ moyen des sept modèles en îles sur les instances de MOTSP

En considérant tout d'abord les résultats sur le MOTSP, les performances de `Clustering original`, `Séparation en cônes` et `APM-MOEA` sont relativement proches avec 2 et 4 îles même si le modèle `APM-MOEA` obtient constamment les plus grandes valeurs d'hypervolume $\mathscr{S}$. Les modèles de séparation de l'espace objectif ont plus de facilité à recouvrir l'espace objectif avec des fronts Pareto fournis en solutions et continus. En augmentant le nombre d'îles, les résultats sont une fois de plus en faveur de `APM-MOEA`. Pour les instances de MQAP, l'analyse de la Figure 4.16 montre que le modèle `APM-MOEA` obtient de meilleures valeurs

d'hypervolume dans toutes les configurations pour ce problème. Ces résultats indiquent que la portion de l'espace objectif qui est dominée par les solutions fournies par `APM-MOEA` est généralement plus grande que les autres modèles ce qui dénote, une nouvelle fois, d'une meilleure convergence du modèle et d'une habileté à mieux explorer l'espace de recherche. Il faut également noter que si, avec la métrique de convergence *IGD*, les modèles `Clustering original` et `DRMOGA` semblaient les plus performants derrière `APM-MOEA`, leur capacité à couvrir l'espace objectif n'est pas confirmée par les valeurs d'hypervolume $\mathscr{S}$ obtenues.

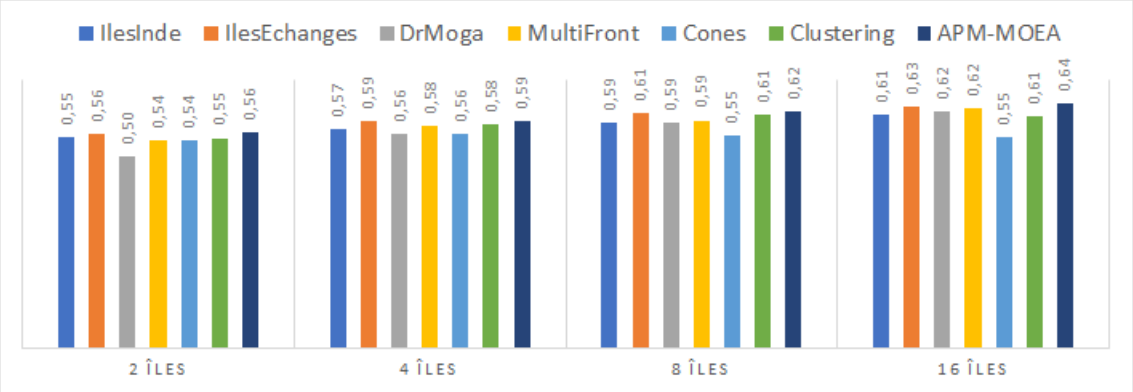


Figure 4.16: Hypervolume $\mathscr{S}$ moyen des sept modèles en îles sur les instances de MQAP

La suite de l'expérimentation est consacrée à l'étude de la diversité des solutions obtenues qui est mesurée par la métrique d'espacement minimal *ms*. Le Tableau 4.7 présente les valeurs d'espacement minimal *ms* moyennes obtenues pour la résolution des problèmes de MOTSP et de MQAP. Plus la valeur de l'espacement minimal *ms* est faible, meilleure est la répartition des solutions de l'ensemble. Dans la résolution du MOTSP, avec n'importe quel nombre d'îles, le modèle `APM-MOEA` fournit une meilleure répartition des solutions que tous les autres modèles comme l'indique les plus petites valeurs d'espacement minimal *ms* obtenues. De plus, au contraire du modèle `Séparation en cônes`, la diversité est améliorée en augmentant le nombre d'îles utilisées. Dans toutes les configurations, les modèles `DRMOGA` et `MFED` obtiennent des valeurs d'espacement minimal qui vacillent entre $0,0073$ et $0,0099$, se révélant à peine meilleurs que le modèle `IlesInde`. En ce qui concerne les instances de MQAP, le modèle `MFED` propose une meilleure distribution des solutions en utilisant seulement deux îles. Cependant, dans les autres configurations, le modèle `APM-MOEA` montre la meilleure capacité à diversifier les ensembles de solutions fournis puisqu'il obtient les plus petites valeurs d'espacement minimal *ms*. L'utilisation d'un algorithme de regroupement amélioré pour distribuer les calculs permet d'obtenir des solutions mieux réparties sur l'espace objectif que les autres modèles et

notamment que le modèle `Clustering original`.

	MOTSP				MQAP			
	2 îles	*4 îles*	*8 îles*	*16 îles*	*2 îles*	*4 îles*	*8 îles*	*16 îles*
`IlesInde`	0,0083	0,0095	0,0085	0,0083	0,0300	0,0293	0,0283	0,0303
`pMOMA`	0,0077	0,0057	0,0045	0,0057	0,0229	0,0241	0,0190	0,0192
`DRMOGA`	0,0086	0,0073	0,0078	0,0090	0,0351	0,0316	0,0290	0,0327
`MFED`	0,0098	0,0095	0,0099	0,0090	**0,0257**	0,0290	0,0265	0,0255
`Séparation en cônes`	0,0047	0,0034	0,0038	0,0059	0,0277	0,0232	0,0229	0,0256
`Clustering original`	0,0038	0,0024	0,0020	0,0025	0,0281	0,0217	0,0193	0,0187
`APM-MOEA`	**0,0037**	**0,0022**	**0,0016**	**0,0015**	0,0266	**0,0212**	**0,0185**	**0,0177**

Tableau 4.7: Espacement minimal *ms* moyen des sept modèles en îles pour les instances de MOTSP et de MQAP

Afin de constater graphiquement l'extensibilité de l'approche et confirmer les résultats des métriques, des exemples de solutions obtenues par `APM-MOEA` dans chacune des configurations sur des instances MQAP à 50 et 100 sites sont proposés aux Figures 4.17 et 4.18. Pour mieux comprendre les résultats proposés, il est nécessaire de rappeler que le MQAP est un problème de minimisation. D'après les deux exemples, la qualité des solutions augmente significativement en termes de convergence en augmentant le nombre d'îles comme en témoignent les fonctions objectif plus petites des solutions obtenues avec 16 îles. Par ailleurs, la répartition des solutions semble meilleure dans les plus grandes configurations en particulier dans la résolution de l'instance à 50 sites illustrée à la Figure 4.17. Ces constatations confirment visuellement les résultats des quatre métriques précédemment présentés quant à l'extensibilité de `APM-MOEA`.

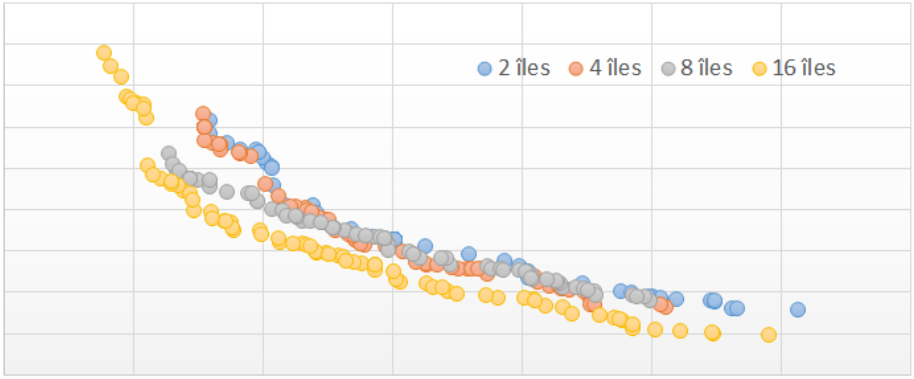


Figure 4.17: Représentation graphique pour la résolution typique de l'instance *Gar50-2fl-1uni* par APM-MOEA en utilisant 2, 4, 8 et 16 îles

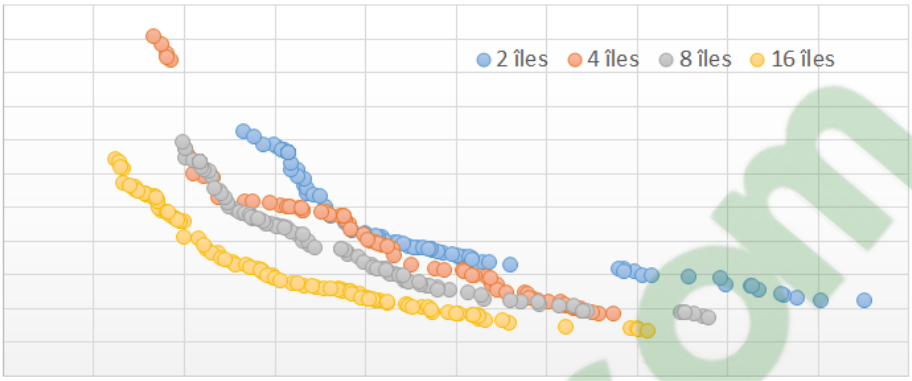


Figure 4.18: Représentation graphique pour la résolution typique de l'instance *Gar100-2fl-2uni* par APM-MOEA en utilisant 2, 4, 8 et 16 îles

4.2.5 TEMPS D'EXÉCUTION DES MODÈLES

La dernière partie de l'expérimentation s'intéresse aux temps d'exécution des modèles en îles implémentés. Le Tableau 4.8 présente les temps d'exécution moyens (en secondes) pour effectuer 100000 générations sur des instances de MOTSP de 100 et 1000 villes et des instances de MQAP de 30 et 100 villes.

	`IlesInde`	`pMOMA`	`DRMOGA`	`MFED`	`Séparation en cônes`	`Clustering original`	`APM-MOEA`
MOTSP							
100	289	310	317	312	325	333	314
1000	713	722	744	722	782	793	731
MQAP							
30	317	323	338	325	343	341	330
100	1700	1711	1732	1712	1735	1741	1715

Tableau 4.8: Temps d'exécution moyens des sept modèles en îles avec l'utilisation de 8 îles pour 100000 générations

D'après les temps répertoriés, il apparaît tout naturellement que c'est le modèle `IlesInde` qui est le plus rapide, puisqu'il ne nécessite pas de calcul supplémentaire et qu'aucune communication n'est nécessaire entre les processeurs. Les modèles `Séparation en cônes` et `Clustering original` engendrent, à l'inverse, les plus grands surcoûts de calcul et de communication comme en témoignent les plus grands temps d'exécution obtenus. Les barrières de synchronisation nécessaires au fonctionnement de ces modèles et les nombreux échanges expliquent ces différences de temps. Les échanges de communication asynchrones dans le modèle `APM-MOEA` permettent de limiter significativement les surcoûts liés au modèle. En effet,

les différences avec le modèle `Clustering original` sont d'autant plus remarquables avec des instances de grande taille puisque ces dernières impliquent des échanges généralement plus importants car contenant plus de données et de nombreuses variables de décision. De même, les temps d'exécution de `APM-MOEA` sont relativement proches des temps des modèles qui demandent peu d'échanges et peu de synchronisations tels que les modèles `MFED` et `pMOMA`.

Finalement, dans le but de constater les gains apportés par l'asynchronisme, un profilage de code a été effectué sur une version synchrone et une version asynchrone du modèle proposé `APM-MOEA`. Les proportions du temps consacrées à l'application et aux opérations MPI sont présentées à la Figure 4.19. Avec des communications synchrones, il apparaît que lorsque le nombre d'îles augmente, les opérations MPI prennent une part de plus en plus importante dans l'exécution du modèle. Les échanges de communications et les barrières de synchronisation occupent jusqu'à 16% du temps d'exécution total avec l'utilisation de 16 îles. À l'inverse, la version asynchrone permet de concentrer la quasi-totalité des efforts sur le modèle en tant que tel puisque dans toutes les configurations la proportion des opérations MPI est inférieure à 1%. Ces différents résultats mettent en avant la nécessité d'utiliser des échanges de communications asynchrones dans un modèle parallèle et plus particulièrement pour APM-MOEA.

Figure 4.19: Profilage du code des versions synchrone et asynchrone de APM-MOEA

4.3 CONCLUSION

La comparaison expérimentale de modèles en îles de la littérature, effectuée au cours de ce chapitre, a été basée sur deux problèmes combinatoires classiques et sur la parallélisation de l'algorithme GISMOO. Les métriques multi-objectifs calculées ont permis de mettre en évidence la capacité de APM-MOEA à mieux converger vers le front Pareto que les autres modèles. Son habilité à trouver des solutions mieux réparties sur l'espace objectif a été également révélée ce qui montre la contribution de l'algorithme de partitionnement et de l'archive globale dans la bonne répartition du travail des îles. Par ailleurs, l'augmentation du nombre d'îles dans le modèle APM-MOEA améliore significativement la qualité de solutions en termes de convergence et de diversité ce qui permet d'attester de l'extensibilité de l'approche. À l'inverse, les modèles parallèles existants ont plus de difficultés a exploiter efficacement de multiples cœurs de processeur. En outre, les gains de temps apportés par les échanges de communication asynchrones ont également été mis en évidence à travers un profilage de code. L'asynchronisme permet de réduire fortement les surcoûts du modèle parallèle et de consacrer les efforts de calculs sur les opérations de l'AEMO.

Bien que l'expérimentation de ce chapitre ait permis d'identifier les qualités du modèle APM-MOEA, les solutions fournies par celui-ci sont encore éloignées des approches conçues spécifiquement pour les problèmes de MOTSP ou de MQAP qui ont été proposées dans la littérature. Afin d'obtenir un modèle compétitif, l'ajout d'une amélioration locale semble indispensable puisque la majorité des algorithmes performants utilisent une forme de recherche locale dans leur processus de résolution. L'expérimentation complète de APM-MOEA avec l'amélioration locale et l'ajout des îles témoins fait l'objet du prochain chapitre.

CHAPITRE 5

EXPÉRIMENTATION ET ÉVALUATION DU COMPORTEMENT DE APM-MOEA POUR LA RÉSOLUTION DE PROBLÈMES COMBINATOIRES

Les problèmes combinatoires représentent une part importante des problèmes réels puisqu'ils couvrent une variété de domaines comme l'ingénierie ou la médecine (Munakata et Barták, 2010). Néanmoins, il a été constaté lors de l'état de l'art des approches parallèles multi-objectifs que ce sont les problèmes à variables continues qui ont bénéficié de la plus grande attention des chercheurs. Comme cette thèse s'intéresse plus particulièrement à l'optimisation combinatoire multi-objectifs, la résolution de deux problèmes combinatoires classiques est réalisée par le modèle APM-MOEA complet, c'est-à-dire avec l'amélioration locale et les îles témoins. L'objectif de ce chapitre est de positionner les résultats de APM-MOEA par rapport à ceux obtenus par les algorithmes spécifiquement conçus pour résoudre les problèmes de MOTSP et de MQAP.

Dans la première partie du chapitre, des instances de MOTSP classiques sont résolues par APM-MOEA et par des modèles de séparation de l'espace objectif sur les bases de la parallélisation de l'algorithme GISMOO (Zinflou *et al.*, 2012). Les expérimentations permettent de comparer les performances des différents modèles implémentés à travers plusieurs métriques multi-objectifs. Les résultats sont également utilisés pour analyser l'extensibilité de l'approche, pour évaluer l'apport des îles témoins au sein du modèle et pour mettre en avant l'importance de l'amélioration locale.

La seconde partie du chapitre est consacrée à l'évaluation du comportement de APM-MOEA dans la résolution du MQAP en comparaison avec d'autres modèles parallèles de la littérature. Les ensembles de solutions obtenus sont également comparés avec ceux fournis par l'algorithme parallèle PasMoQAP (Sanhueza *et al.*, 2017). De plus, la performance de APM-

MOEA est testée sur des instances de grande taille produites par le générateur de Knowles et Corne (2003).

5.1 RÉSOLUTION DU PROBLÈME MOTSP

5.1.1 AMÉLIORATION LOCALE POUR LE MOTSP

Pour améliorer localement les solutions du MOTSP au sein du modèle APM-MOEA, le voisinage $2-opt$ a été choisi. En plus de sa relative simplicité, il présente l'avantage de permettre un calcul rapide des fonctions objectif du voisin généré à partir des fonctions objectif de la solution initiale. En considérant une instance symétrique, le mouvement $2-opt$ consiste à supprimer deux arêtes d'un tour et à reconnecter les sommets afin de recréer un chemin complet. La Figure 5.1 présente un exemple de voisin généré avec un $2-opt$ pour un problème à huit villes. À partir d'une permutation $1-2-3-4-5-6-7-8$, deux arêtes sont retirées $(2,3)$ et $(6,7)$ et deux autres $(2-6)$ et $(3-7)$ sont ajoutées afin de compléter le tour. La nouvelle solution devient alors $1-2-6-5-4-3-7-8$

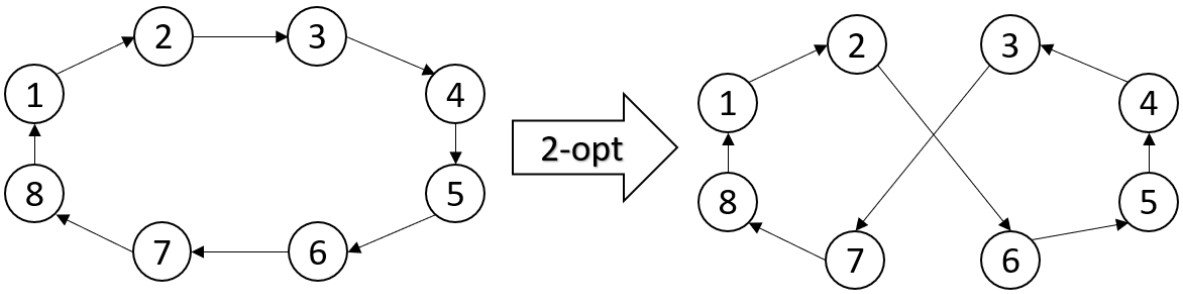


Figure 5.1: Exemple de voisinage $2-opt$ avec huit villes

Le voisinage $2-opt$ d'une solution S correspond à l'ensemble des solutions obtenues en appliquant un mouvement $2-opt$ sur S. Avec cette définition, la taille du voisinage d'une solution est égale à $N(N-3)/2$ où N représente le nombre de villes. Dans le modèle APM-MOEA, la stratégie par liste de candidats est employée (Algorithme 7) afin de diminuer les efforts de calcul. L'approche utilisée dans l'algorithme 2PPLS (Lust et Teghem, 2010c) est adaptée pour générer les candidats à partir des solutions contenues dans l'archive locale de chaque île. Pour chaque arête (i,j), qui est contenue dans au moins une des solutions de l'archive, la ville j est ajoutée à la liste des candidats de i. Le pseudo-code présenté à l'Algorithme 14 permet de comprendre comment est généré le voisinage d'une solution

$S = (\pi_1, \pi_2, ..., \pi_N)$ à partir de la liste des candidats préalablement calculée. Ce code est intégré dans APM-MOEA pour améliorer localement les solutions du MOTSP. Finalement, pour calculer les fonctions objectif de la solution voisine S', il suffit de conserver les valeurs de S et de retirer les distances des deux arêtes retirées puis d'ajouter les distances des deux nouvelles arêtes.

Algorithme 14: Pseudo-code de la génération du voisinage $2-opt$

```
Données: N, nombre de villes
         Solution S = (π_1, π_2, ...., π_N)
Pour i allant de 1 à N Faire
    S' = S;
    Supprimer l'arête (π_i, π_{i+1}) dans S' ;
    Pour Chaque k inclus dans la liste des candidats de π_{i+1} Faire
        Ajouter l'arête (π_{i+1}, k) à S';
        Reconnecter les sommets restants;
        Traitement de S';
    FinPour
FinPour
```

5.1.2 DESCRIPTION DE L'EXPÉRIMENTATION

L'étude expérimentale du modèle proposé pour la résolution du MOTSP est divisée en quatre parties. Dans la première partie, deux versions du modèle APM-MOEA sont confrontées : une implémentation avec les îles témoins et l'autre sans celles-ci. Cette comparaison permet d'identifier l'apport des îles témoins dans les performances du modèle. La deuxième partie met en avant la contribution de l'amélioration locale au sein de APM-MOEA par le biais d'une nouvelle comparaison de deux versions du modèle : une avec l'activation de l'amélioration locale et l'autre sans. La troisième partie a pour but de comparer les performances de APM-MOEA avec les trois autres modèles qui se sont révélés les plus performants dans le chapitre précédent sur les bases de la parallélisation de GISMOO : l'algorithme `DRMOGA` (Hiroyasu *et al.*, 2000), l'approche `Séparation en cônes` (Branke *et al.*, 2004) et le modèle `Clustering original` (Streichert *et al.*, 2005). Afin de présenter une comparaison juste et équitable, l'amélioration locale est incorporée dans chacun des modèles avec un même niveau d'intensification. Finalement, dans la quatrième partie, une analyse de l'évolution de la qualité des solutions au cours du temps est effectuée pour souligner l'extensibilité de APM-MOEA.

5.1.3 CONDITIONS DE L'EXPÉRIMENTATION

L'expérimentation a été une nouvelle fois effectuée sur le supercalculateur ROMEO (ROMEO, 2018) avec 2, 4, 8 et 16 cœurs de processeur. Dans l'ensemble des modèles distribués, chaque île est associée à un unique cœur de processeur. En suivant les recommandations des auteurs de GISMOO (Zinflou *et al.*, 2012), la taille de la population est fixée à 100 et le taux de mutation à $0,05$. Pour faire évoluer les populations, le croisement OX (Davis, 1985) et la mutation par inversion (Larrañaga *et al.*, 1999) ont été choisis. L'amélioration locale est appliquée toutes les *nGen* générations, où *nGen* a été fixé empiriquement à 20 de sorte à équilibrer l'intensification et la diversification. Les instances présentées à la Section 4.1.1.2 et contenant de 200 à 1000 villes sont résolues dans le cadre de cette expérimentation. Les temps de résolution maximum, notés *tMax*, accordés à chacune des instances et les intervalles de migration *iMig* sont présentés en secondes dans le Tableau 5.1.

Les métriques utilisées dans le chapitre précédent sont réutilisées pour comparer les performances des modèles parallèles. En outre, les solutions fournies par Lust et Teghem (2010c) permettent de constituer les fronts Pareto approximés. Chaque instance a été résolue 10 fois par les modèles parallèles `APM-MOEA`, `DRMOGA`, `Séparation en cônes` et `Clustering original` et les moyennes des métriques ont été retenues.

Instance	*tMax*	*iMig*	Instance	*tMax*	*iMig*	Instance	*tMax*	*iMig*
kroAB200	200	3	EuclAB300	300	4	MixedAB300	300	4
kroAB300	300	4	EuclAB500	1000	10	MixedAB500	1000	10
kroAB400	600	5	RdAB300	300	4			
kroAB500	1000	10	RdAB500	1000	10			
kroAB750	2500	20	ClusteredAB300	300	4			
kroAB1000	5000	30	ClusteredAB500	1000	10			

Tableau 5.1: Instances de MOTSP résolues et les temps *tMax* et *iMig* en secondes

5.1.4 APPORT DES ÎLES TÉMOINS

Dans cette première partie de l'expérimentation, la contribution des îles témoins dans le modèle APM-MOEA est analysée. Pour ce faire, deux versions du modèle `APM-T` et `APM-C` sont directement comparées. Dans la première version, les îles témoins sont incluses dans le modèle alors que `APM-C` utilise seulement des îles classiques. Le Tableau 5.2 expose les

moyennes normalisées de la métrique de convergence *IGD* pour chacune des instances pour le modèle APM-MOEA avec et sans îles témoins. Pour chaque configuration selon le nombre d'îles, les meilleurs résultats sont indiqués en gras.

	4 îles		8 îles		16 îles	
	APM-C	APM-T	APM-C	APM-T	APM-C	APM-T
kroAB200	**0,0033**	0,0039	**0,0025**	0,0029	0,0021	**0,0020**
kroAB300	**0,0050**	0,0077	**0,0031**	0,0047	0,0032	**0,0031**
kroAB400	**0,0057**	0,0083	0,0042	**0,0040**	0,0032	**0,0029**
kroAB500	**0,0066**	0,0104	**0,0039**	0,0054	0,0035	**0,0028**
kroAB750	**0,0083**	0,0084	0,0059	**0,0056**	0,0037	**0,0034**
kroAB1000	**0,0072**	0,0073	**0,0056**	0,0065	0,0045	**0,0036**
EuclAB300	**0,0035**	0,0037	0,0024	**0,0023**	0,0021	**0,0012**
EuclAB500	0,0039	**0,0034**	0,0020	**0,0016**	0,0015	**0,0013**
RdAB300	**0,0103**	0,0196	0,0071	**0,0070**	0,0067	**0,0050**
RdAB500	**0,0106**	0,0192	0,0110	**0,0080**	0,0069	**0,0057**
ClusteredAB300	**0,0027**	0,0054	0,0017	**0,0014**	0,0013	**0,0012**
ClusteredAB500	0,0030	**0,0027**	0,0028	**0,0020**	0,0013	**0,0011**
MixedAB300	**0,0054**	0,0076	0,0034	**0,0029**	0,0033	**0,0029**
MixedAB500	**0,0039**	0,0079	0,0046	**0,0036**	0,0034	**0,0021**
Moyenne	**0,0057**	0,0082	0,0043	**0,0041**	0,0033	**0,0027**

Tableau 5.2: Convergence *IGD* moyenne pour les versions APM-C et APM-T avec 4, 8 et 16 îles sur des instances de MOTSP

Les résultats pour la métrique de convergence *IGD* montrent que l'ajout des îles témoins dans un modèle avec peu d'îles est inefficace. En effet, APM-C obtient des solutions plus proches du front Pareto pour la majorité des instances en utilisant 4 îles comme en témoignent les valeurs de convergence *IGD* plus faibles. En exploitant 8 cœurs de processeur, les résultats sont plus proches puisque sur 4 des 14 instances APM-C obtient une meilleure valeur de convergence *IGD* et sur les 10 autres c'est la variante APM-T qui obtient de meilleurs résultats. Les valeurs moyennes sur toutes les instances sont également proches ($0,0043$ contre $0,0041$), même si les îles témoins permettent d'obtenir des solutions légèrement plus proches du front Pareto. En outre, pour les modèles avec 16 îles, la variante APM-T surclasse plus significativement le modèle APM-C. Sur toutes les instances résolues, l'obtention de valeurs de convergence *IGD* plus proches de 0 démontre une meilleure faculté à trouver des solutions avoisinant le front Pareto.

La bonne convergence du modèle APM-T est confirmée par la métrique d'hypervolume dont les résultats sont exposés à la Figure 5.2. Les plus grandes valeurs d'hypervolume $\mathscr{S}$ obtenues par APM-T dans les configurations à 8 et 16 îles montrent que les îles témoins permettent à APM-MOEA de fournir des solutions qui couvrent mieux l'espace objectif. Par ailleurs, la diversité des solutions est étudiée en se basant sur les résultats de la métrique d'espacement minimal *ms* qui sont présentés au Tableau 5.3. Similairement à la métrique de convergence *IGD*, APM-C obtient les meilleures valeurs d'espacement minimal *ms* avec 4 îles pendant que APM-T fournit des solutions mieux espacées avec 8 et 16 îles. Ces résultats confirment que les îles témoins améliorent la capacité de diversification et d'exploration de APM-MOEA, plus particulièrement avec un grand nombre d'îles. Par conséquent, dans la suite de ces expérimentations, les îles témoins sont conservées dans le modèle APM-MOEA pour les configurations avec plus de quatre îles.

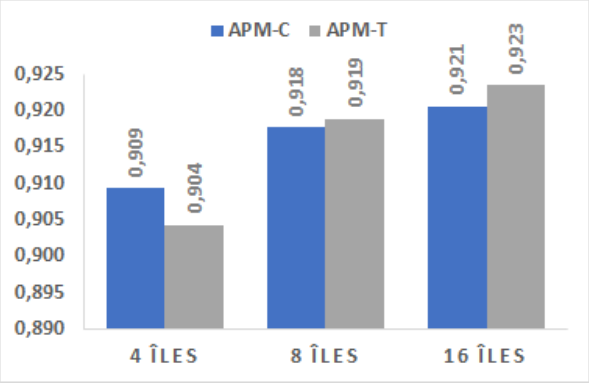


Figure 5.2: Hypervolume $\mathscr{S}$ moyen des versions APM-C et APM-T sur les instances de MOTSP

	APM-C	APM-T
4 îles	**0,00039**	0,00050
8 îles	0,00023	**0,00022**
16 îles	0,00019	**0,00014**

Tableau 5.3: Espacement minimal *ms* moyen des versions APM-C et APM-T sur les instances de MOTSP

5.1.5 CONTRIBUTION DE L'AMÉLIORATION LOCALE

La deuxième partie d'expérimentation a pour but de mettre en évidence l'importance de l'amélioration locale (AL) au sein du modèle APM-MOEA et dans des approches multi-objectifs en général. Pour ce faire, le modèle a été exécuté sans l'amélioration locale, mais en laissant des temps d'exécution maximum identiques à ceux utilisés pour la version avec amélioration locale. Les valeurs de la métrique de convergence *IGD* des deux versions du modèle sont exposées au Tableau 5.4 pour toutes les instances de MOTSP. Pour la version sans amélioration locale, les deux îles témoins agissent de la même manière et génèrent des solutions sans les améliorer localement.

	4 îles		8 îles		16 îles	
	Avec AL	Sans AL	Avec AL	Sans AL	Avec AL	Sans AL
kroAB200	**0,0039**	0,0108	**0,0029**	0,0059	**0,0020**	0,0041
kroAB300	**0,0077**	0,0195	**0,0047**	0,0099	**0,0031**	0,0072
kroAB400	**0,0083**	0,0217	**0,0040**	0,0143	**0,0029**	0,0077
kroAB500	**0,0104**	0,0276	**0,0054**	0,0157	**0,0028**	0,0135
kroAB750	**0,0084**	0,0391	**0,0056**	0,0112	**0,0034**	0,0107
kroAB1000	**0,0073**	0,0479	**0,0065**	0,0173	**0,0036**	0,0099
EuclAB300	**0,0037**	0,0222	**0,0023**	0,0179	**0,0012**	0,0151
EuclAB500	**0,0034**	0,0661	**0,0016**	0,0328	**0,0013**	0,0297
RdAB300	**0,0196**	0,0720	**0,0070**	0,0090	**0,0050**	0,0058
RdAB500	**0,0192**	0,0954	**0,0080**	0,0100	**0,0057**	0,0092
ClusteredAB300	**0,0054**	0,0152	**0,0014**	0,0121	**0,0012**	0,0117
ClusteredAB500	**0,0027**	0,0175	**0,0020**	0,0172	**0,0011**	0,0127
MixedAB300	**0,0076**	0,0192	**0,0029**	0,0237	**0,0029**	0,0188
MixedAB500	**0,0079**	0,0294	**0,0036**	0,0264	**0,0021**	0,0147
Moyenne	**0,0082**	0,0360	**0,0041**	0,0160	**0,0027**	0,0122

Tableau 5.4: Convergence *IGD* moyenne de APM-MOEA avec et sans amélioration locale (AL) pour les instances de MOTSP

D'après les résultats présentés pour la métrique de convergence *IGD*, il apparaît que l'amélioration locale permet d'améliorer significativement la convergence des solutions puisque les distances aux fronts Pareto sont largement diminuées. Cette tendance est vérifiée sur tous les types d'instance et notamment sur celles de plus grande taille. Les valeurs moyennes d'hypervolume sont également présentées dans la Figure 5.3 et mettent une nouvelle fois en avant l'intérêt de la recherche locale dans une approche de résolution multi-objectifs. En effet, dans chacune des configurations, les valeurs d'hypervolume $\mathscr{S}$ sont supérieures pour le modèle APM-MOEA avec l'amélioration locale. Des différences de 0.40 à 0.057 sont constatées et dénotent une capacité supérieure du modèle avec AL à couvrir l'espace objectif et à converger vers le front Pareto.

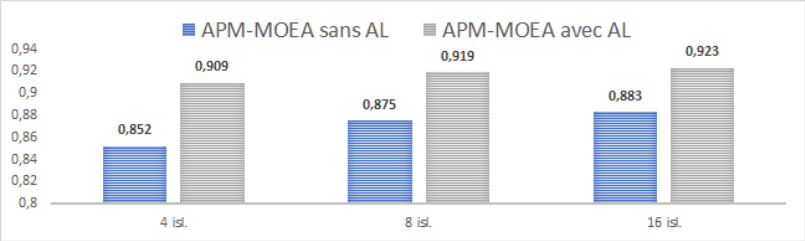


Figure 5.3: Hypervolume $\mathscr{S}$ moyen de APM-MOEA avec et sans amélioration locale (AL) pour les instances de MOTSP

Afin de vérifier graphiquement l'importance de l'amélioration locale dans le modèle APM-MOEA, deux exemples d'ensembles de solutions obtenus par l'approche sont proposés aux Figures 5.4 et 5.5. Pour les deux instances, les solutions de APM-MOEA sans AL sont relativement éloignées des solutions du modèle avec AL comme l'avaient indiqué les valeurs de convergence *IGD* précédemment présentées. De plus, les solutions sont moins bien réparties dans l'espace objectif que les autres ensembles exposés ce qui est le plus remarquable dans la résolution de l'instance *MixedAB300* (Figure 5.5). L'amélioration locale permet à la fois d'intensifier les recherches mais aussi d'ajouter une capacité exploratoire importante au modèle. Enfin, les deux exemples présentés permettent de constater la bonne convergence du modèle APM-MOEA avec AL puisque celui-ci obtient des solutions très proches des fronts Pareto obtenus grâce à l'algorithme 2PPLS (Lust et Teghem, 2010c) qui a été spécialement conçu pour la résolution du MOTSP. Les distances entre les solutions de APM-MOEA avec AL et du front Pareto de 2PPLS sont relativement réduites comme l'ont montré les faibles valeurs de convergence *IGD*. Les solutions obtenues par APM-MOEA avec AL semblent dans certaines zones confondues avec celles du front Pareto approximé.

Les résultats de la comparaison de APM-MOEA avec AL et sans AL ont mis en avant l'importance d'un mécanisme de recherche locale pour améliorer significativement la qualité des solutions. Ainsi, dans la suite de cette expérimentation, le modèle APM-MOEA inclura ce mécanisme ainsi que les îles témoins qui ont été prouvées performantes dans la précédente section.

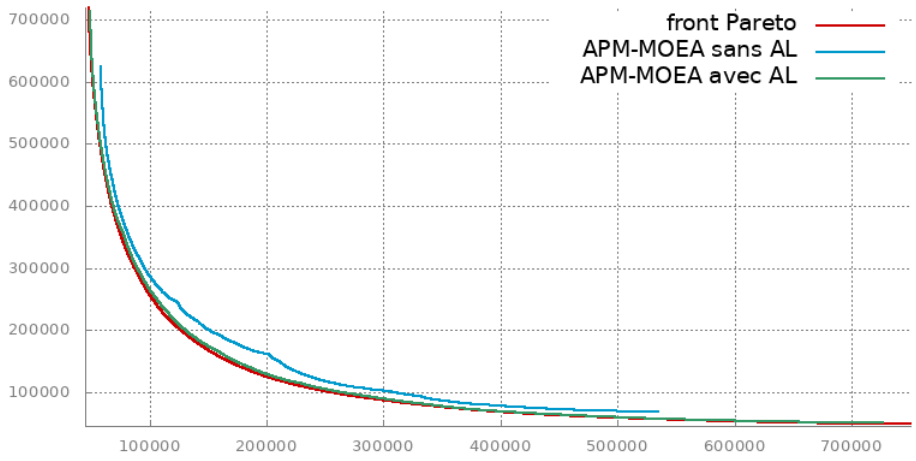


Figure 5.4: Représentation graphique pour la résolution typique de l'instance *kroAB500* par APM-MOEA avec et sans amélioration locale (AL) en utilisant 16 îles

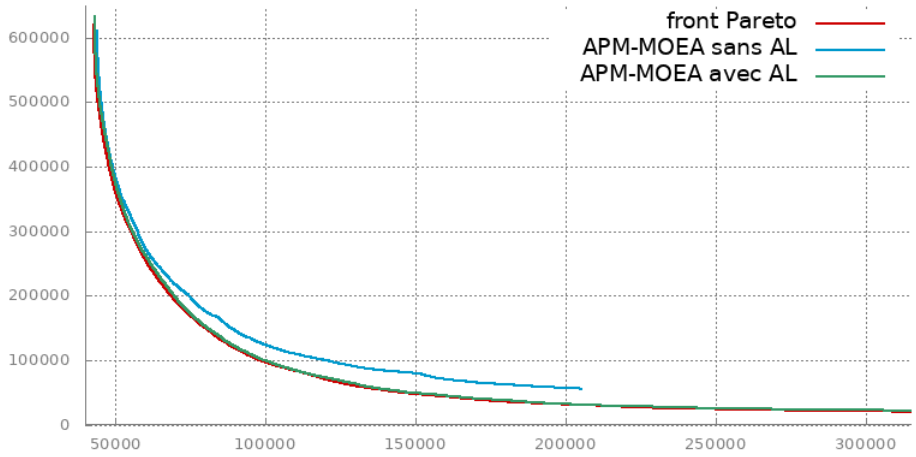


Figure 5.5: Représentation graphique pour la résolution typique de l'instance *MixedAB300* par APM-MOEA avec et sans amélioration locale (AL) en utilisant 16 îles

5.1.6 COMPARAISON AUX MODÈLES DE SÉPARATION DE L'ESPACE OBJECTIF

Cette troisième partie de l'expérimentation compare la qualité des solutions non-dominées obtenue par APM-MOEA et les trois modèles parallèles de la littérature : `DRMOGA`, `Séparation en cônes`, `Clustering original`. Les Tableaux 5.5 et 5.6 présentent les moyennes des valeurs de convergence *IGD* en utilisant 2, 4, 8 et 16 îles. Pour le calcul de la métrique, les fonctions objectif ont été normalisées en utilisant le point Nadir et le point Idéal. Pour chacune des configurations selon le nombre d'îles, la meilleure valeur obtenue est marquée en gras.

2 îles	DRMOGA	Séparation en cônes	Clustering original	APM-MOEA
kroAB200	0,0056	**0,0044**	0,0071	0,0045
kroAB300	0,0092	0,0096	0,0083	**0,008**
kroAB400	0,0105	0,0098	0,0086	**0,0084**
kroAB500	0,0102	**0,0086**	0,0103	0,0103
kroAB750	0,0104	**0,0085**	0,0128	0,0126
kroAB1000	0,0108	**0,0087**	0,0113	0,0113
EuclAB300	0,0085	**0,0047**	0,0072	0,0061
EuclAB500	0,0073	**0,0053**	0,0073	0,0057
RdAB300	0,0236	0,0201	0,0198	**0,0182**
RdAB500	0,0273	0,0196	0,0245	**0,0174**
ClusteredAB300	0,0050	0,0044	0,0048	**0,0040**
ClusteredAB500	0,0061	0,0054	0,0056	**0,0048**
MixedAB300	0,0090	0,0082	0,0086	**0,0071**
MixedAB500	0,0099	**0,0076**	0,0082	0,0097

4 îles	DRMOGA	Séparation en cônes	Clustering original	APM-MOEA
kroAB200	0,0039	0,0050	0,0038	**0,0033**
kroAB300	0,0065	0,0062	0,0058	**0,0050**
kroAB400	0,0058	0,0064	0,0061	**0,0057**
kroAB500	0,0073	0,0065	**0,0055**	0,0066
kroAB750	**0,0070**	0,0072	0,0080	0,0083
kroAB1000	0,0084	0,0078	0,0076	**0,0072**
EuclAB300	0,0041	0,0041	0,0039	**0,0035**
EuclAB500	0,0041	**0,0034**	0,0047	0,0039
RdAB300	0,0147	0,0179	0,0110	**0,0103**
RdAB500	0,0155	0,0170	0,0137	**0,0106**
ClusteredAB300	0,0039	0,0041	0,0028	**0,0027**
ClusteredAB500	**0,0029**	0,0034	0,0035	0,0030
MixedAB300	0,0059	0,0064	0,0055	**0,0054**
MixedAB500	0,0051	0,0055	0,0049	**0,0039**

Tableau 5.5: Convergence *IGD* **moyenne de** DRMOGA**,** Séparation en cônes**,** Clustering original **et** APM-MOEA **sur les instances de MOTSP en utilisant 2 et 4 îles**

8 îles	DRMOGA	Séparation en cônes	Clustering original	APM-MOEA
kroAB200	0,0035	0,0040	**0,0022**	0,0029
kroAB300	0,0040	0,0045	**0,0036**	0,0047
kroAB400	0,0058	0,0048	0,0041	**0,004**
kroAB500	0,0058	0,0048	**0,0006**	0,0043
kroAB750	0,0058	0,0058	0,0056	**0,0052**
kroAB1000	0,0062	0,0060	**0,0057**	0,0065
EuclAB300	0,0029	0,0030	0,0049	**0,0023**
EuclAB500	0,0025	0,0021	0,0057	**0,0016**
RdAB300	0,0090	0,0196	0,0199	**0,0070**
RdAB500	0,0107	0,0223	0,0158	**0,008**
ClusteredAB300	0,0031	0,0042	0,0029	**0,0014**
ClusteredAB500	0,0039	0,0035	0,0035	**0,0020**
MixedAB300	0,0052	0,0069	0,0061	**0,0029**
MixedAB500	**0,0035**	0,0058	0,0061	0,0036

16 îles	DRMOGA	Séparation en cônes	Clustering original	APM-MOEA
kroAB200	0,0021	0,0046	0,0020	**0,0019**
kroAB300	0,0037	0,0052	0,0035	**0,0031**
kroAB400	0,0043	0,0053	0,0041	**0,0029**
kroAB500	0,0049	0,0054	0,0037	**0,0028**
kroAB750	0,0057	0,0058	0,0039	**0,0034**
kroAB1000	0,0062	0,0061	0,0041	**0,0036**
EuclAB300	0,0042	0,0050	0,0041	**0,0012**
EuclAB500	0,0024	0,0036	0,0039	**0,0013**
RdAB300	0,0069	0,0459	0,0153	**0,0050**
RdAB500	0,0134	0,0324	0,0163	**0,0057**
ClusteredAB300	0,0022	0,0044	0,0027	**0,0012**
ClusteredAB500	0,0020	0,0031	0,0024	**0,0011**
MixedAB300	0,0046	0,0101	0,0065	**0,0029**
MixedAB500	0,0036	0,0102	0,0084	**0,0021**

Tableau 5.6: Convergence *IGD* **moyenne de** DRMOGA**,** Séparation en cônes**,** Clustering original **et** APM-MOEA **sur les instances de MOTSP en utilisant 8 et 16 îles**

D'après les résultats de cette métrique, en utilisant 2 îles, il apparaît que le modèle `Séparation en cônes` trouve de meilleures solutions que les autres modèles sur la moitié des instances résolues. Pour les autres instances, c'est le modèle `APM-MOEA` qui propose une meilleure convergence au front Pareto. Avec un nombre plus élevé d'îles, `APM-MOEA` surclasse les autres modèles implémentés, plus particulièrement avec 16 îles où il obtient une meilleure valeur de convergence *IGD* sur toutes les instances de MOTSP. En outre, pour le modèle APM-MOEA, les valeurs de convergence *IGD* diminuent significativement dans les plus grandes configurations, ce qui montre sa capacité à améliorer la qualité de ses solutions lorsque le nombre d'îles augmente. Au contraire, le modèle `Séparation en cônes` ne parvient pas à trouver de meilleures solutions avec l'utilisation d'un grand nombre d'îles. Bien que tous les modèles parallèles implémentés intègrent une amélioration locale avec le même niveau d'intensification, APM-MOEA parvient à trouver des solutions plus proches du front Pareto que les autres modèles grâce notamment à une meilleure répartition des îles dans l'espace objectif.

Pour confirmer les résultats de la bonne convergence de APM-MOEA, la métrique de couverture $\mathscr{C}$ de deux ensembles a été utilisée pour comparer `APM-MOEA` par rapport aux autres modèles de la littérature. Les Figures 5.6 et 5.7 présentent les résultats des modèles `APM-MOEA`, `Séparation en cônes`, `Clustering original` et `DRMOGA` selon la métrique de couverture $\mathscr{C}$ pour respectivement des modèles à 8 et 16 îles qui permettent de fournir les meilleurs ensembles de solutions.

Les résultats présentés à la Figure 5.6 favorisent significativement le modèle `APM-MOEA`. En effet, les valeurs moyennes de couverture $\mathscr{C}$(`APM-MOEA`, `DRMOGA`), $\mathscr{C}$(`APM-MOEA`, `Séparation en cônes`) et $\mathscr{C}$(`APM-MOEA`, `Clustering original`) sont plus grandes que les valeurs de $\mathscr{C}$(`DRMOGA`, `APM-MOEA`), $\mathscr{C}$(`Séparation en cônes`, `APM-MOEA`) et $\mathscr{C}$(`Clustering original`, `APM-MOEA`) pour toutes les instances résolues. Néanmoins, les modèles de la littérature parviennent à trouver une faible proportion de solutions qui ne sont pas dominées par `APM-MOEA` comme en attestent les valeurs de couverture $\mathscr{C}$(`DRMOGA`, `APM-MOEA`) qui peuvent atteindre $0,30$ pour certaines instances.

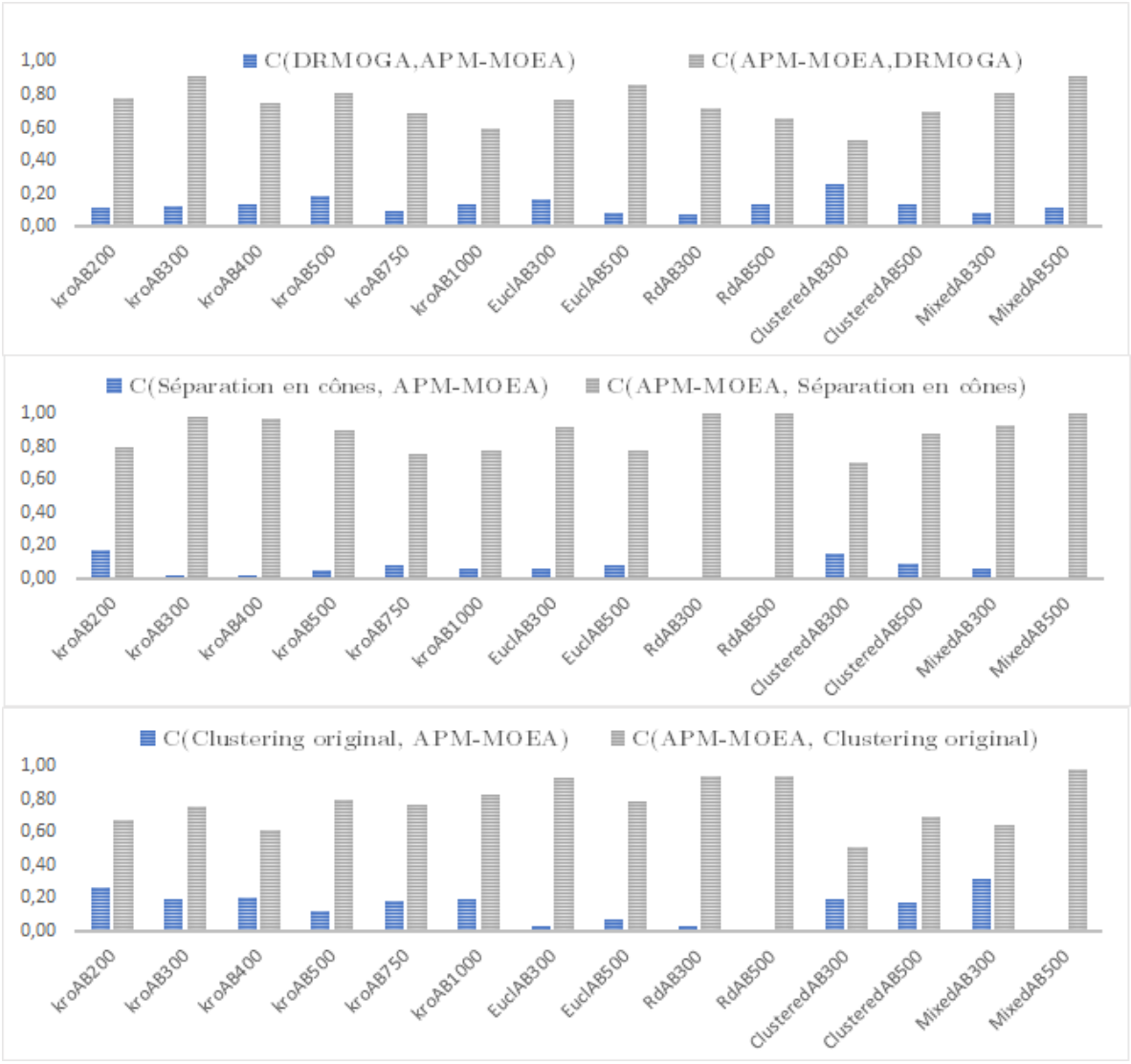


Figure 5.6: Couverture $\mathscr{C}$ moyenne de `APM-MOEA` par rapport à `DRMOGA`, `Séparation en cônes` et `Clustering original` en utilisant 8 îles sur les instances de MOTSP

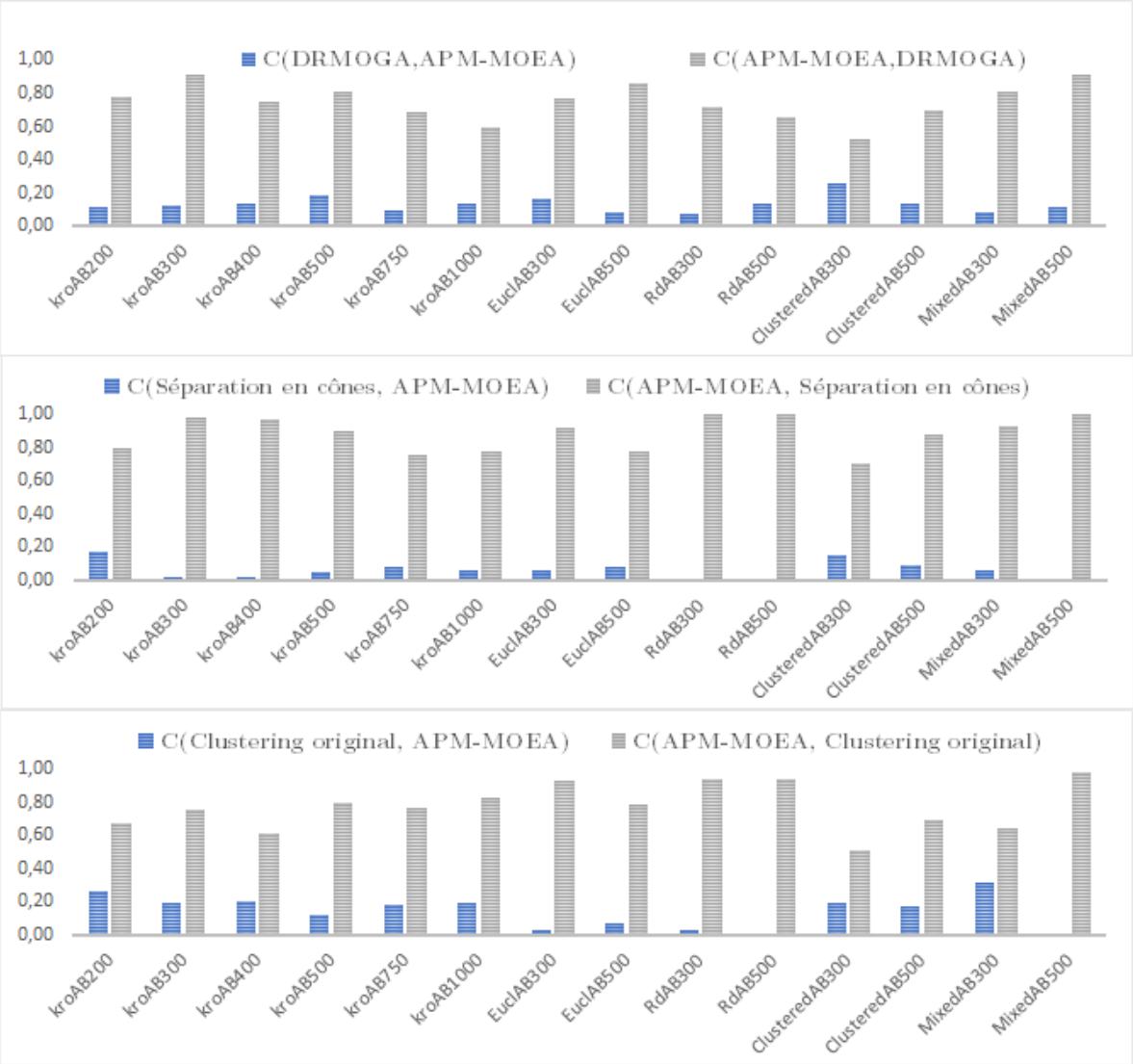


Figure 5.7: Couverture $\mathscr{C}$ moyenne de `APM-MOEA` par rapport à `DRMOGA`, `Séparation en cônes` et `Clustering original` en utilisant 16 îles sur les instances de MOTSP

Avec l'utilisation de 16 îles (Figure 5.7), la supériorité de APM-MOEA sur les autres modèles est amplifiée. Les valeurs moyennes de $\mathscr{C}$(`APM-MOEA`, `Séparation en cônes`) varient entre $0,80$ et $1,00$ et indiquent que la majorité des solutions trouvées par le modèle `Séparation en cônes` sont dominées par des solutions du modèle `APM-MOEA`. Au contraire, les valeurs de $\mathscr{C}$(`Séparation en cônes`, `APM-MOEA`) proches de 0 montrent que le modèle APM-MOEA trouve très peu de solutions qui sont dominées par celles de `Séparation en cônes`. Les modèles `Clustering original` et `DRMOGA` réussissent, quant à eux, à trouver des solutions d'une meilleure qualité que `Séparation en cônes` comme en témoignent

les valeurs de $\mathscr{C}$(`Clustering original`, `APM-MOEA`) et $\mathscr{C}$(`DRMOGA`, `APM-MOEA`) qui sont, en moyenne, faiblement supérieures à celles de $\mathscr{C}$(`Séparation en cônes`, `APM-MOEA`). Encore une fois et de manière plus manifeste, la plupart des solutions trouvées par `APM-MOEA` ne sont pas dominées par celles de `Clustering original` et `DRMOGA` avec l'utilisation de 16 îles.

Les métriques de convergence *IGD* et de couverture $\mathscr{C}$ de deux ensembles ont permis d'attester de la capacité du modèle `APM-MOEA` à mieux converger vers le front Pareto que les modèles de la littérature. Pour confirmer cette tendance et mettre en évidence la distribution des solutions dans l'espace objectif, la métrique d'hypervolume $\mathscr{S}$ est utilisée. Il peut être nécessaire de rappeler qu'elle permet d'estimer la portion de l'espace objectif qui est dominée par un ensemble de solutions fourni. Les valeurs moyennes de $\mathscr{S}$ obtenues sur toutes les instances par APM-MOEA et les trois modèles de la littérature sont présentées à la Figure 5.8.

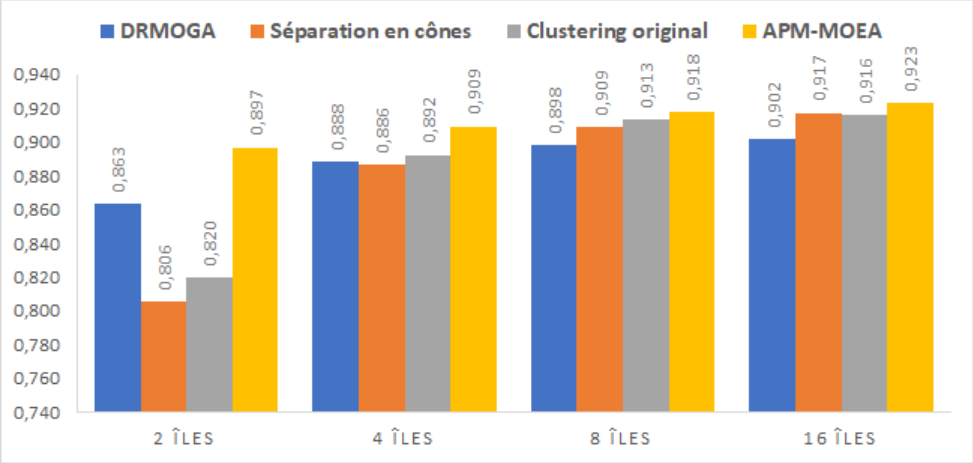


Figure 5.8: Hypervolume $\mathscr{S}$ moyen de `DRMOGA`, `Séparation en cônes`, `Clustering original` et `APM-MOEA` sur les instances de MOTSP

Dans chacune des configurations, le modèle `APM-MOEA` obtient des valeurs d'hypervolume $\mathscr{S}$ plus grandes que les autres modèles. Encore une fois, les différences les plus importantes sont constatées avec un nombre d'îles élevé. Ces résultats montrent que APM-MOEA fournit des solutions qui couvrent de manière plus importante l'espace objectif. Il est également intéressant de noter que, si le modèle `DRMOGA` montre une relative bonne convergence selon la métrique de convergence *IGD*, il ne réussit pas à proposer une bonne distribution de ses solutions comme l'indiquent les résultats de la métrique d'hypervolume $\mathscr{S}$. Les valeurs moyennes d'hypervolume $\mathscr{S}$ obtenues par `APM-MOEA`, combinées aux résultats des deux premières métriques, précisent la capacité de `APM-MOEA` à fournir des solutions plus proches

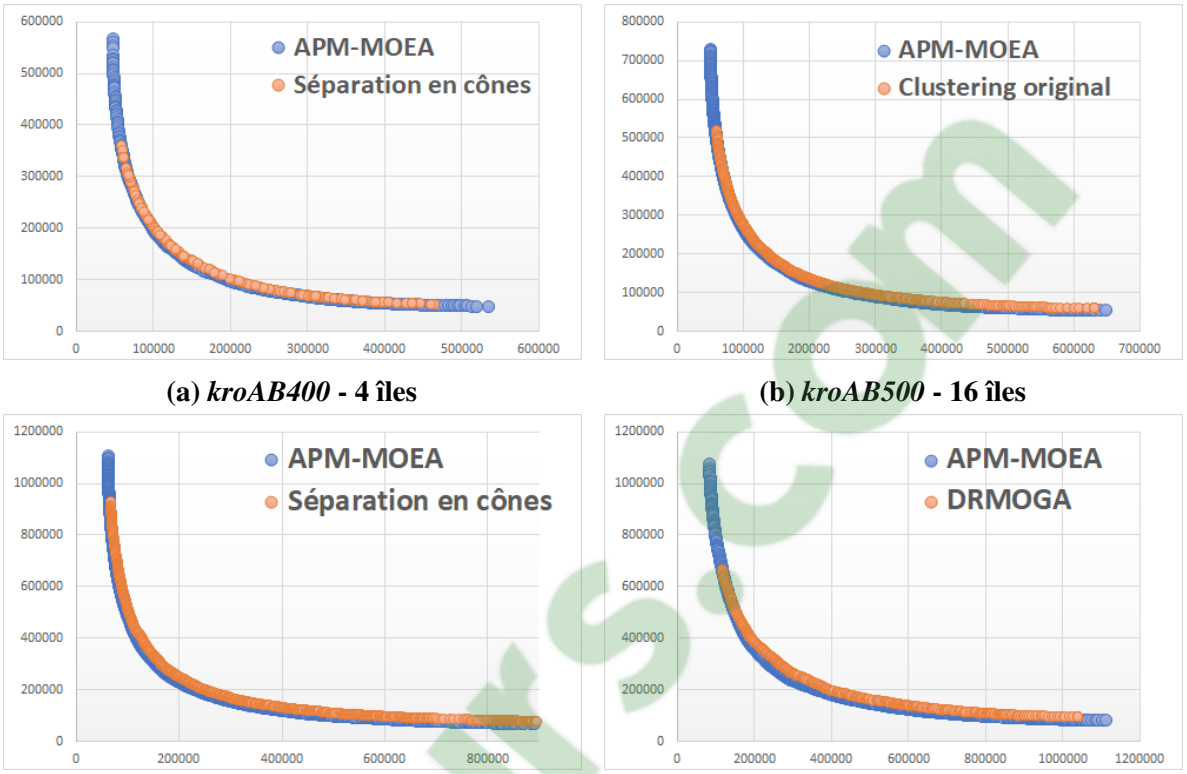
du front Pareto que les autres modèles. Elles présagent également de la bonne distribution des solutions obtenues. Pour vérifier cette affirmation, la métrique d'espacement minimal *ms* a été calculée et permet d'analyser exclusivement la diversité des solutions. Le Tableau 5.7 résume les valeurs moyennes d'espacement minimal *ms* obtenues par les modèles `DRMOGA`, `Séparation en cônes`, `Clustering original` et `APM-MOEA` sur toutes les instances en fonction du nombre d'îles.

	2 îles	4 îles	8 îles	16 îles
`DRMOGA`	0,00388	0,00237	0,00167	0,00145
`Séparation en cônes`	0,00160	0,00061	0,00033	0,00030
`Clustering original`	0,00135	0,00046	0,00034	0,00032
`APM-MOEA`	**0,00064**	**0,00039**	**0,00022**	**0,00014**

Tableau 5.7: Espacement minimal *ms* moyen de `DRMOGA`, `Séparation en cônes`, `Clustering original` et `APM-MOEA` sur les instances de MOTSP

Selon les valeurs d'espacement minimal *ms*, le modèle `APM-MOEA` obtient les plus petites valeurs dans n'importe quelle configuration ce qui prouve sa faculté à fournir une meilleure distribution des solutions que les modèles de la littérature. Par ailleurs, la diversité s'améliore avec l'augmentation du nombre d'îles. Par le biais des fonctionnalités de APM-MOEA présentées dans le troisième chapitre, comme l'algorithme de regroupement amélioré ou la vue globale donnée à l'île organisatrice, il parvient à fournir des solutions mieux réparties sur l'espace objectif. Les îles témoins qui n'ont pas de contrainte sur les zones de recherche permettent également d'améliorer la capacité d'exploration du modèle.

Pour terminer la phase de comparaison, la Figure 5.9 illustre les résultats des quatre métriques en proposant des exemples de solutions obtenues lors d'une exécution typique des différents modèles dans la résolution des instances de MOTSP. Les sous-titres de chaque graphique indiquent les noms des instances résolues et le nombre d'îles employées. Ces résultats confirment que le modèle APM-MOEA fournit des solutions qui sont mieux distribuées et qui couvrent une région plus large de l'espace objectif confirmant les résultats des métriques d'hypervolume $\mathscr{S}$ et d'espacement minimal *ms*. Ces exemples mettent également en évidence la bonne convergence du modèle `APM-MOEA` qui obtient des valeurs de fonctions objectif plus faibles.



(a) ***kroAB400*** **- 4 îles** **(b)** ***kroAB500*** **- 16 îles**

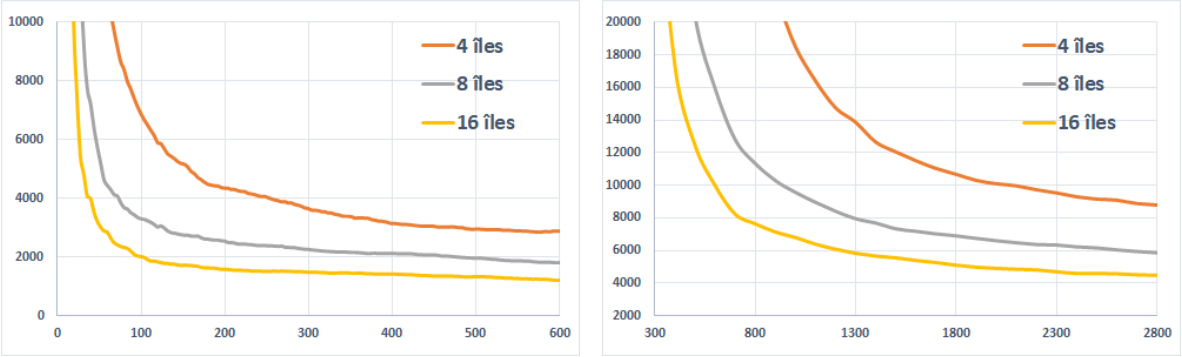
(c) ***kroAB750*** **- 16 îles** **(d)** ***kroAB1000*** **- 8 îles**

Figure 5.9: Représentation graphique pour la résolution typique de 4 instances de MOTSP par APM-MOEA et les trois modèles parallèles de la littérature

5.1.7 ÉVOLUTION DE LA QUALITÉ DES SOLUTIONS EN FONCTION DU NOMBRE D'ÎLES

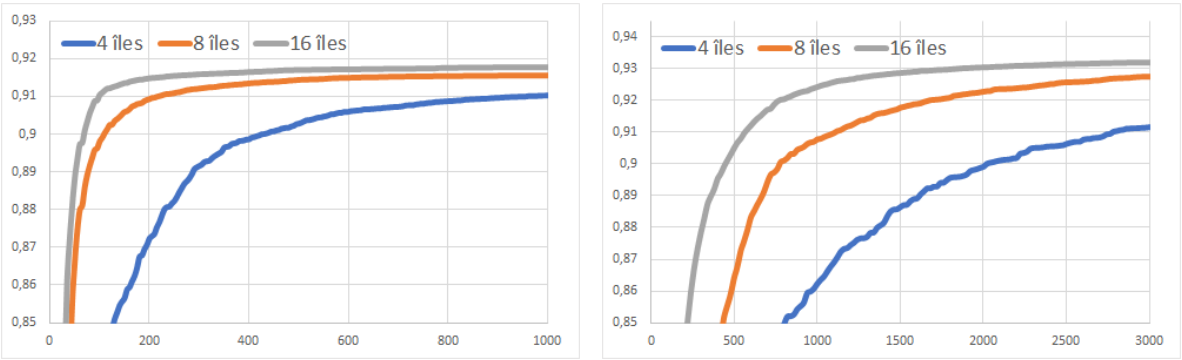
Les métriques calculées lors de phase précédente ont permis de mettre en évidence qu'en augmentant le nombre d'îles, la qualité des solutions obtenues par le modèle APM-MOEA est significativement améliorée. Cette dernière partie de l'expérimentation a pour but de fournir plus de précisions à propos de l'extensibilité du modèle, c'est-à-dire sa capacité à améliorer la qualité des solutions avec le nombre d'îles. Pour ce faire, l'analyse de l'évolution de la qualité des solutions de l'archive globale au cours de l'exécution est effectuée sur les différentes configurations. L'archive globale a été périodiquement enregistrée par l'île organisatrice dans des fichiers externes et les valeurs des métriques ont été calculées par la suite pour éviter des coûts de calcul supplémentaires. La Figure 5.10 présente l'évolution de la métrique de convergence *IGD* en utilisant 4, 8 et 16 îles pour les instances *kroAB400* et *kroAB750*.

Dans les deux exemples exposés, un grand nombre d'îles permet d'atteindre une meilleure convergence vers le front Pareto approximé et de manière plus rapide. En effet, la configuration avec 16 îles prend beaucoup moins de temps que les deux autres pour atteindre une valeur de *IGD* spécifique. Les plus grands écarts sont observés entre le modèle à 4 îles et le modèle à 8 îles. De la même manière, la Figure 5.11 montre l'évolution de l'hypervolume $\mathscr{S}$ pour les différentes configurations sur les deux mêmes instances de problème.



(a) ***kroAB400*** **(b)** ***kroAB750***

Figure 5.10: Évolution de la métrique de convergence *IGD* au cours du temps (en secondes) pour l'archive globale de `APM-MOEA` sur deux instances de MOTSP



(a) ***kroAB400*** **(b)** ***kroAB750***

Figure 5.11: Évolution de la métrique d'hypervolume $\mathscr{S}$ au cours du temps (en secondes) pour l'archive globale de `APM-MOEA` sur deux instances de MOTSP

Encore une fois, la configuration à 16 îles permet d'obtenir de plus grandes valeurs d'hypervolume $\mathscr{S}$ ce qui souligne une meilleure répartition des solutions fournies et une plus grande couverture de l'espace objectif. Les plus grands écarts sont également constatés entre 4 et 8 îles. Finalement, les résultats sur l'hypervolume et la métrique d'espacement minimal

ont montré la capacité de APM-MOEA à bien répartir la recherche du front Pareto sur les différentes îles dans les grandes configurations.

5.1.8 CONCLUSION SUR LA RÉSOLUTION DU MOTSP

La phase d'expérimentation étendue du modèle APM-MOEA dans la résolution du MOTSP a permis de mettre en évidence les qualités de l'approche proposée. Sur un jeu de 14 instances de test, le modèle a montré ses facultés à mieux converger vers le front Pareto que des modèles de séparation de l'espace objectif qui ont été proposés dans la littérature. La répartition des îles amenée par la méthode de regroupement améliorée et le maintien d'une archive globale permettent d'explorer de manière plus efficace l'espace de recherche. La capacité de APM-MOEA à fournir des solutions de meilleure qualité en augmentant le nombre d'îles a également été identifiée et indique une bonne extensibilité du modèle. En outre, les îles témoins permettent à APM-MOEA de fournir des ensembles de solutions diversifiés et bien répartis dans l'espace objectif. Finalement, même si le modèle parallèle appliqué à l'algorithme GISMOO ne parvient pas à atteindre les performances d'algorithme spécialisé dans la résolution du MOTSP tel que 2PPLS (Lust et Teghem, 2010c), APM-MOEA permet de s'approcher des meilleurs résultats. En effet, les solutions obtenues par APM-MOEA sont, par rapport aux autres modèles comparés, les plus proches des fronts Pareto approximés.

5.2 RÉSOLUTION DU PROBLÈME MQAP

5.2.1 AMÉLIORATION LOCALE POUR LE MQAP

Pour résoudre le MQAP à l'aide du modèle APM-MOEA, il a été nécessaire d'adapter l'amélioration locale aux caractéristiques du problème. Tout comme pour le MOTSP, les solutions de MQAP sont représentées sous la forme d'une permutation d'éléments ce qui permet de préciser les affectations des unités aux différents sites disponibles. Parmi les transformations locales existantes sur ce type de représentation, c'est la transposition (ou *swap*) qui a été choisie pour le modèle. Elle correspond à un échange de deux éléments au sein de la solution.

Pour calculer le coût d'une transposition, la méthode proposée par Taillard (1995) pour le QAP a été adaptée pour la version multi-objectifs du problème. Ainsi, à partir d'une permutation π, la différence Δ obtenue en échangeant les éléments $\pi(i)$ et $\pi(j)$ pour la fonction objectif f est calculée comme définit à l'Équation 5.1.

$$\Delta(\pi,i,j)^f = \sum_{\substack{k=0 \\ k\neq i,j}}^{n} \Big((a_{ki}-a_{kj})(b^f_{\pi_k\pi_j}-b^f_{\pi_k\pi_i}) + (a_{ik}-a_{jk})(b^f_{\pi_j\pi_k}-b^f_{\pi_i\pi_k}) \Big) \tag{5.1}$$

où n est le nombre d'unités, a_{ij} est la distance entre les sites i et j, $b^k_{\pi_i\pi_j}$ est le coût du k-ième flux entre l'unité π_i et l'unité π_j, et π_i correspond au i-ème élément dans la permutation π.

Il est nécessaire de noter que ce calcul est valable lorsque la matrice des distances est symétrique ce qui est le cas pour toutes les instances de MQAP traitées durant cette thèse. Cette méthode permet d'accélérer l'évaluation des fonctions objectif d'un voisin obtenu par une transposition. Pour le modèle APM-MOEA, la stratégie de *first improvement* est utilisée et génère tout le voisinage d'une solution jusqu'à obtenir une solution qui domine la solution courante. Pour ce faire, toutes les transpositions sont testées dans un ordre défini et en commençant par une position aléatoire r. Ainsi, les $n-1$ premières transpositions seront (π_r,π_{r+1}) puis (π_r,π_{r+2}) jusqu'à (π_r,π_{r+n-1}). Si aucune solution dominante n'est trouvée, alors il faut partir de la positon $r+1$ et réaliser les transpositions $(\pi_{r+1},\pi_{r+2}),(\pi_{r+1},\pi_{r+3}),...,(\pi_{r+1},\pi_{r+n-2})$. Ces étapes sont répétées jusqu'à tester toutes les transformations possibles ou jusqu'à obtenir une solution qui domine la solution courante.

5.2.2 DESCRIPTION DE L'EXPÉRIMENTATION

L'expérimentation de cette section a pour but de mettre en exergue les performances du modèle dans la résolution du MQAP. Dans une première partie, le jeu d'instances à 60 variables de décision proposé par Garrett et Dasgupta (2009)(voir Section 4.1.1.1) a été utilisé pour une première comparaison de APM-MOEA avec des modèles de séparation de l'espace objectif : l'algorithme `DRMOGA` (Hiroyasu *et al.*, 2000), l'approche `Séparation en cônes` (Branke *et al.*, 2004) et le modèle `Clustering original` (Streichert *et al.*, 2005). Ces instances ont également été résolues par le modèle parallèle PasMoQAP (Sanhueza *et al.*, 2017) et les ensembles de solutions obtenus par celui-ci ont été rendus disponibles par les auteurs. La deuxième partie de l'expérimentation a donc pour but de comparer la qualité des solutions obtenues par APM-MOEA et par PasMoQAP. Enfin, afin d'évaluer les performances du modèle proposé dans un plus large contexte, des instances de grande taille ont été générées et seront utilisées pour effectuer une nouvelle comparaison dans la dernière partie de l'expérimentation.

5.2.3 CONDITIONS DE L'EXPÉRIMENTATION

L'algorithme GISMOO (Zinflou *et al.*, 2012) a une nouvelle fois été parallélisé selon les modèles `APM-MOEA`, `DRMOGA`, `Séparation en cônes` et `Clustering original`. Comme pour la résolution du MOTSP, la taille de la population est fixée à 100 et le taux de mutation à $0,05$. Pour une comparaison juste, tous les modèles comparés ont été implémentés avec une amélioration locale telle que décrite dans la Section 5.2.1. L'expérimentation a été effectuée sur le supercalculateur ROMEO (ROMEO, 2018) avec 2, 4, 8 et 16 cœurs de processeur.

Les métriques de convergence *IGD*, d'hypervolume $\mathscr{S}$, de couverture de deux ensembles $\mathscr{C}$ et la mesure d'espacement minimal *ms* sont utilisées pour évaluer les performances des modèles. Les moyennes des métriques obtenues lors de 10 exécutions de chaque modèle sont présentées pour toutes les instances. Pour générer les fronts de référence utilisés pour la métrique *IGD*, les solutions non-dominées trouvées par tous les modèles implémentés ainsi que les solutions fournies par PasMoQAP (Sanhueza *et al.*, 2017) sont utilisées.

5.2.4 COMPARAISON AUX MODÈLES DE SÉPARATION DE L'ESPACE OBJECTIF

Cette section est consacrée à la comparaison empirique du modèle APM-MOEA et des modèles parallèles DRMOGA, Séparation en cônes et Clustering original. En se basant sur les travaux de Sanhueza *et al.* (2017), le temps d'exécution maximum pour chaque instance est de 200 secondes. La première métrique de convergence *IGD* permet de mesurer exclusivement la convergence des solutions au front Pareto. Les valeurs moyennes obtenues par les 4 modèles sont présentées dans les Tableaux 5.8, 5.9 et 5.10 pour chacune des configurations selon le nombre d'îles. Il faut noter que dans le but d'obtenir des valeurs moyennes représentatives pour toutes les instances, les fonctions objectif ont été normalisées à partir du point Idéal et du point Nadir.

2 îles	DRMOGA	Séparation en cônes	Clustering original	APM-MOEA
Gar60-2fl-1uni	0,0144	0,0130	0,0089	**0,0084**
Gar60-2fl-2uni	0,0197	0,0197	0,0240	**0,0183**
Gar60-2fl-3uni	0,0472	0,0345	**0,0227**	0,0378
Gar60-2fl-4uni	0,0046	0,0030	**0,0027**	0,0030
Gar60-2fl-5uni	0,0147	0,0163	0,0171	**0,0147**
Gar60-2fl-1rl	0,0097	0,0088	0,0063	**0,0056**
Gar60-2fl-2rl	0,0081	**0,0053**	0,0074	0,0067
Gar60-2fl-3rl	0,0089	0,0131	0,0110	**0,0069**
Gar60-2fl-4rl	0,0108	0,0138	0,0078	**0,0055**
Gar60-2fl-5rl	0,0282	0,0370	0,0257	**0,0245**
Gar60-3fl-1uni	0,0239	**0,0198**	0,0215	0,0226
Gar60-3fl-2uni	0,0333	0,0251	**0,0227**	0,0237
Gar60-3fl-3uni	0,0432	0,0401	0,0406	**0,0347**
Gar60-3fl-1rl	0,0273	**0,0153**	0,0192	0,0228
Gar60-3fl-2rl	0,0286	**0,0179**	0,0195	0,0187
Gar60-3fl-3rl	0,0221	0,0184	0,0274	**0,0182**

Tableau 5.8: Convergence *IGD* moyenne de DRMOGA, Séparation en cônes, Clustering original et APM-MOEA sur les instances de MQAP à 60 unités en utilisant 2 îles

D'après les résultats exposés au Tableau 5.8, il apparaît que APM-MOEA a une meilleure capacité de convergence moyenne que les autres modèles pour la configuration à 2 îles. En fait, sur 9 des 16 instances de test, le modèle obtient les valeurs de convergence *IGD* les plus faibles.

4 îles	DRMOGA	Séparation en cônes	Clustering original	APM-MOEA
Gar60-2fl-1uni	0,0120	0,0079	0,0068	**0,0052**
Gar60-2fl-2uni	0,0131	0,0164	0,0113	**0,0061**
Gar60-2fl-3uni	0,0234	0,0334	0,0209	**0,0172**
Gar60-2fl-4uni	0,0082	**0,002**	0,0023	0,0022
Gar60-2fl-5uni	0,0151	0,0163	0,0158	**0,0143**
Gar60-2fl-1rl	0,0116	0,0089	0,0091	**0,0059**
Gar60-2fl-2rl	0,0122	0,0078	0,0074	**0,0059**
Gar60-2fl-3rl	0,0150	0,0099	0,0103	**0,0072**
Gar60-2fl-4rl	0,0153	0,0069	0,0066	**0,0058**
Gar60-2fl-5rl	0,0236	0,0307	0,0302	**0,0196**
Gar60-3fl-1uni	0,0184	**0,0142**	0,0212	0,0160
Gar60-3fl-2uni	0,0174	0,0206	**0,0155**	0,0161
Gar60-3fl-3uni	0,0340	0,0475	0,0348	**0,0249**
Gar60-3fl-1rl	0,0230	0,0151	0,0158	**0,015**
Gar60-3fl-2rl	0,0233	0,0151	0,0158	**0,015**
Gar60-3fl-3rl	0,0220	0,0131	**0,0113**	0,0154
Moyenne	0,0173	0,0170	0,0149	**0,0115**

Tableau 5.9: Convergence *IGD* moyenne de DRMOGA, Séparation en cônes, Clustering original et APM-MOEA sur les instances de MQAP à 60 unités en utilisant 4 îles

En utilisant 4 et 8 îles (Tableaux 5.9 et 5.10), les valeurs de convergence *IGD* de APM-MOEA sont plus proches de 0 pour la majorité des instances et indiquent que cette implémentation minimise les distances aux fronts Pareto dans l'espace objectif. Pour la configuration à 16 îles, dont les résultats sont exposés aux Tableau 5.10, APM-MOEA obtient les meilleures moyennes de convergence *IGD* pour toutes les instances. L'augmentation du nombre d'îles permet à APM-MOEA d'améliorer la qualité de ses solutions. Les autres modèles ont, quant à eux, plus de difficultés à atteindre un même niveau de performance comme le montrent les valeurs de convergence *IGD* obtenues avec 16 îles qui sont similaires à celles de APM-MOEA avec 2 îles.

8 îles	DRMOGA	Séparation en cônes	Clustering original	APM-MOEA
Gar60-2fl-1uni	0,0088	0,0091	0,0066	**0,0057**
Gar60-2fl-2uni	0,0111	0,0159	0,0112	**0,0083**
Gar60-2fl-3uni	0,0249	0,0402	0,0201	**0,0197**
Gar60-2fl-4uni	0,0097	0,0021	0,0024	**0,0016**
Gar60-2fl-5uni	0,0156	0,0148	0,0124	**0,0067**
Gar60-2fl-1rl	0,0182	**0,0071**	0,0083	0,0093
Gar60-2fl-2rl	0,0064	0,0133	0,0087	**0,0032**
Gar60-2fl-3rl	0,0108	0,0127	0,0079	**0,0071**
Gar60-2fl-4rl	0,0128	0,0100	0,0075	**0,0044**
Gar60-2fl-5rl	0,0360	0,0350	0,0365	**0,027**
Gar60-3fl-1uni	0,0203	0,0124	0,0167	**0,0112**
Gar60-3fl-2uni	0,0123	0,0156	0,0155	**0,0084**
Gar60-3fl-3uni	0,0195	0,0434	0,0326	**0,0167**
Gar60-3fl-1rl	0,0093	0,0079	0,0121	**0,0064**
Gar60-3fl-2rl	0,0159	**0,011**	0,0110	0,0128
Gar60-3fl-3rl	0,0124	0,0129	0,0107	**0,0087**
Moyenne	0,0154	0,0171	0,0142	**0,0093**

16 îles	DRMOGA	Séparation en cônes	Clustering original	APM-MOEA
Gar60-2fl-1uni	0,0156	0,0099	0,0047	**0,004**
Gar60-2fl-2uni	0,0090	0,0215	0,0086	**0,0042**
Gar60-2fl-3uni	0,0116	0,0372	0,0150	**0,0087**
Gar60-2fl-4uni	0,0170	0,0025	0,0016	**0,0009**
Gar60-2fl-5uni	0,0250	0,1873	0,0998	**0,0165**
Gar60-2fl-1rl	0,0171	0,0089	0,0054	**0,0042**
Gar60-2fl-2rl	0,0098	0,0114	0,0071	**0,0037**
Gar60-2fl-3rl	0,0137	0,0090	0,0052	**0,0046**
Gar60-2fl-4rl	0,0224	0,0106	0,0087	**0,0035**
Gar60-2fl-5rl	0,0159	0,0277	0,0238	**0,0147**
Gar60-3fl-1uni	0,0123	0,0139	0,0161	**0,0083**
Gar60-3fl-2uni	0,0212	0,0087	0,0155	**0,0043**
Gar60-3fl-3uni	0,0304	0,0409	0,0162	**0,0116**
Gar60-3fl-1rl	0,0104	0,0074	0,0083	**0,0059**
Gar60-3fl-2rl	0,0137	0,0087	0,0061	**0,0044**
Gar60-3fl-3rl	0,0207	0,0112	0,0074	**0,0045**
Moyenne	0,0165	0,0283	0,0168	**0,0061**

Tableau 5.10: Convergence *IGD* **moyenne de** DRMOGA**,** Séparation en cônes**,** Clustering original **et** APM-MOEA **sur les instances de MQAP à 60 unités en utilisant 8 et 16 îles**

Afin de comparer directement les ensembles de solutions obtenus par les différents modèles et confirmer les premiers résultats, la métrique de couverture $\mathscr{C}$ de deux ensembles a été utilisée. Les solutions fournies par `APM-MOEA` ont été confrontées à celles de `DRMOGA`, `Séparation en cônes` et `Clustering original` et les résultats de couverture $\mathscr{C}$ sont présentés aux Figures 5.12 et 5.13 pour respectivement les configurations à 8 et 16 îles.

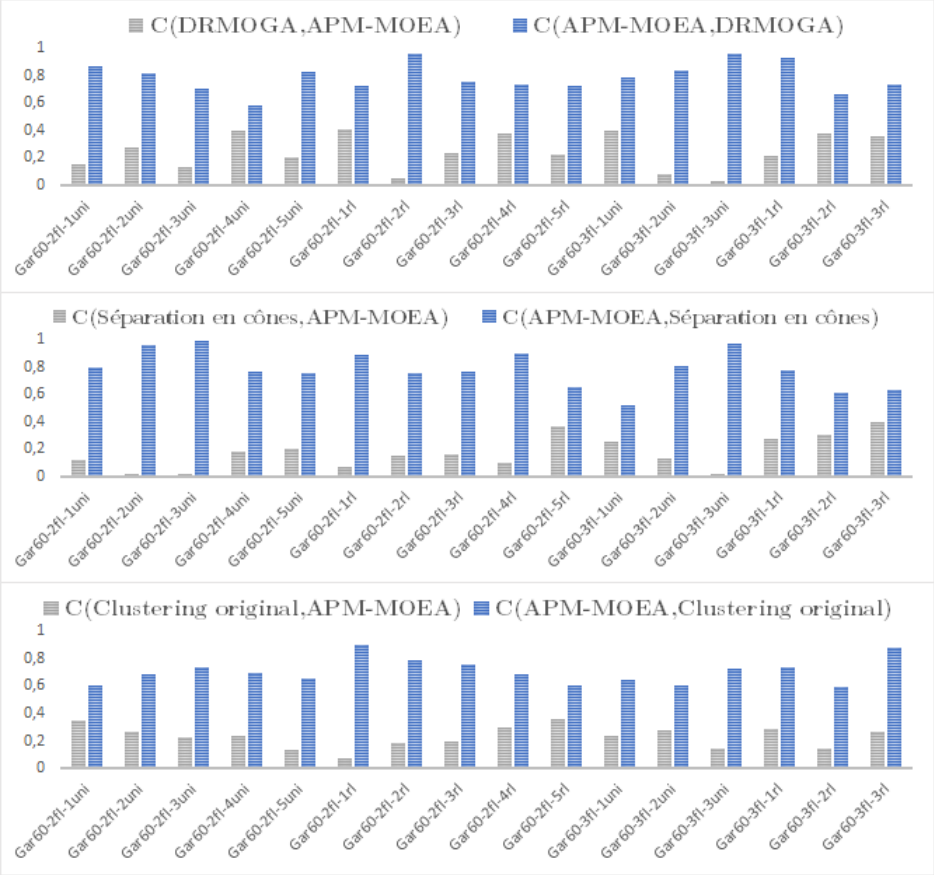


Figure 5.12: Couverture $\mathscr{C}$ moyenne de `APM-MOEA` par rapport à `DRMOGA`, `Séparation en cônes` et `Clustering original` en utilisant 8 îles sur les instances de MQAP à 60 unités

Selon les couvertures $\mathscr{C}$ moyennes, le modèle `APM-MOEA` surclasse une nouvelle fois les autres modèles. En effet, les valeurs moyennes de $\mathscr{C}$(`APM-MOEA`,`DRMOGA`), $\mathscr{C}$(`APM-MOEA`, `Séparation en cônes`) et $\mathscr{C}$(`APM-MOEA`,`Clustering original`) sont comprises entre $0,6$ et 1 pour toutes les instances ce qui indique que la majorité des solutions fournies par les modèles de la littérature sont dominées par au moins une solution obtenue par `APM-MOEA`. À l'inverse, les valeurs faibles de $\mathscr{C}$(`DRMOGA`,`APM-MOEA`), $\mathscr{C}$(`Séparation en cônes`,`APM-MOEA`) et $\mathscr{C}$(`Clustering original`,`APM-MOEA`) confirment la moins bonne convergence des autres

modèles puisqu'ils trouvent peu de solutions qui ne sont pas dominées par celles de APM-MOEA. La supériorité de ce dernier est accentuée avec l'utilisation de 16 îles comme le montrent, par exemple, les valeurs de couverture $\mathscr{C}$(APM-MOEA,Séparation en cônes) très proches de 1 et a contrario les moyennes de $\mathscr{C}$(Séparation en cônes,APM-MOEA) quasiment nulles.

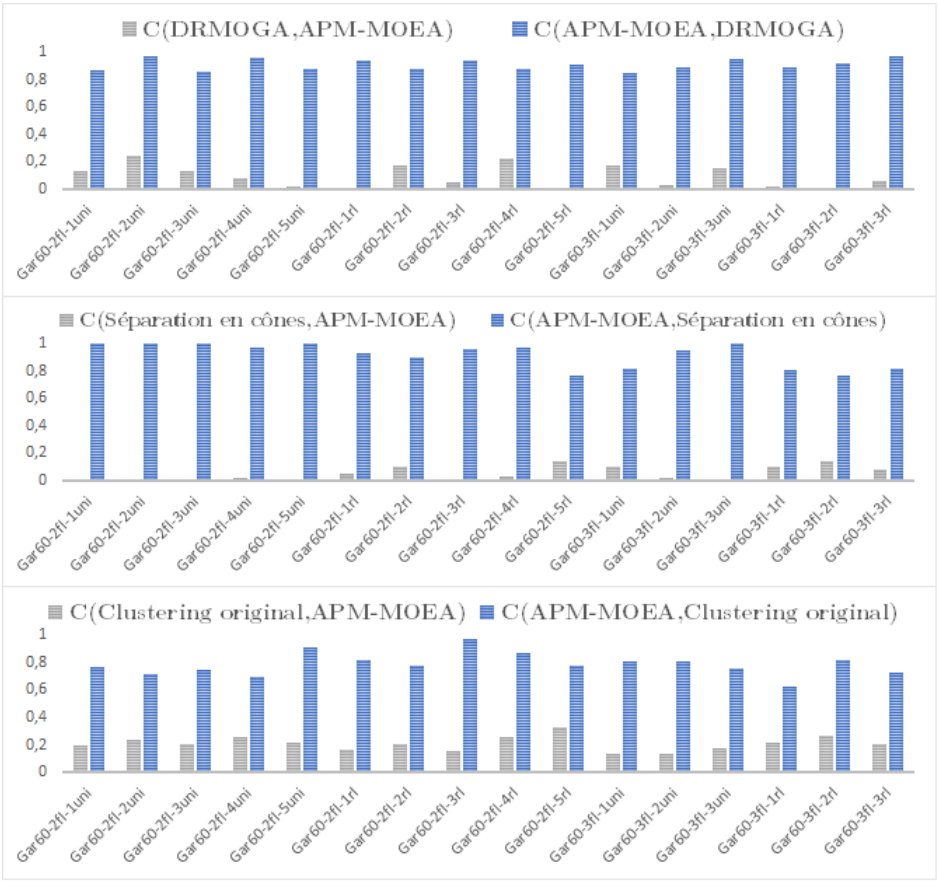


Figure 5.13: Couverture $\mathscr{C}$ moyenne APM-MOEA par rapport à DRMOGA, Séparation en cônes et Clustering original en utilisant 16 îles sur les instances de MQAP à 60 unités

La capacité de convergence du modèle proposé APM-MOEA a été mise en évidence par le biais des deux premières métriques. Par la suite, l'analyse de la distribution des solutions dans l'espace objectif est réalisée à travers deux autres métriques : la métrique d'hypervolume $\mathscr{S}$ et la mesure d'espacement minimal *ms*. Les résultats moyens de la première sont exposés à la Figure 5.14 pour chacun des modèles et dans toutes les configurations.

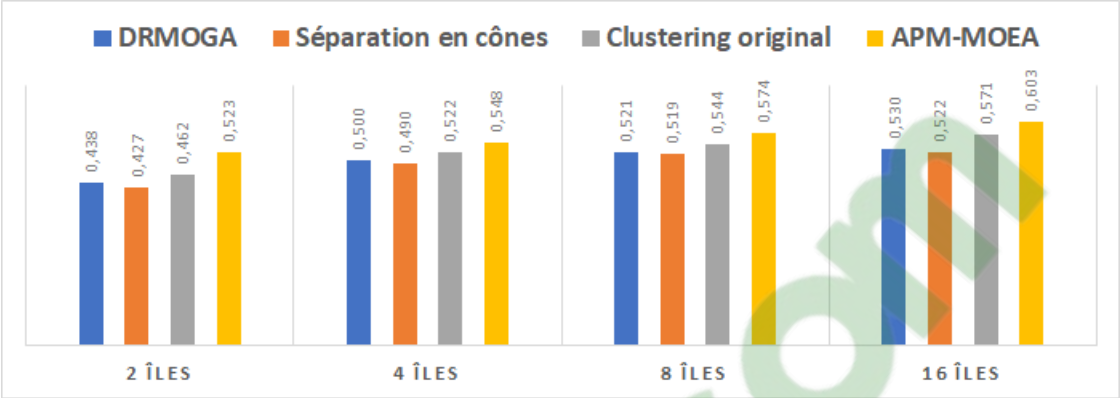


Figure 5.14: Hypervolume $\mathscr{S}$ moyen de `DRMOGA`, `Séparation en cônes`, `Clustering original` et `APM-MOEA` sur les instances de MQAP à 60 unites

D'après les valeurs d'hypervolume $\mathscr{S}$, le modèle `APM-MOEA` se révèle plus performant que les modèles de la littérature. En fait, pour chacune des configurations, il obtient des mesures d'hypervolume $\mathscr{S}$ significativement supérieures aux autres ce qui dénote une habilité à mieux couvrir l'espace objectif. La capacité du modèle `APM-MOEA` à fournir un ensemble de solutions qui couvre de manière plus importante l'espace objectif est confirmée par cette métrique, ce qui permet également d'attester de sa faculté à converger mais aussi à diversifier ses solutions. Pour vérifier cette dernière observation, la mesure d'espacement minimal a été calculée pour tous les modèles et toutes les configurations. Les valeurs moyennes d'espacement minimal *ms* sont exposées au Tableau 5.11.

	2 îles	4 îles	8 îles	16 îles
DRMOGA	0,0211	0,0177	0,0710	0,0650
Séparation en cônes	0,0274	0,0199	0,0174	0,0167
Clustering original	0,0219	0,0162	0,0144	0,0130
APM-MOEA	**0,0158**	**0,0149**	**0,0134**	**0,0124**

Tableau 5.11: Espacement minimal *ms* moyen de `DRMOGA`, `Séparation en cônes`, `Clustering original` et `APM-MOEA` sur les instances de MQAP à 60 unités

Comme l'indiquait la métrique d'hypervolume $\mathscr{S}$, la capacité d'exploration de l'espace objectif du modèle `APM-MOEA` est confirmée par les valeurs d'espacement minimal *ms* obtenues. Pour chaque configuration, le nouveau modèle propose une meilleure distribution des solutions comme en témoignent les plus petites valeurs de *ms* acquises. À l'instar des trois premières

métriques, la métrique d'espacement minimal *ms* indique que APM-MOEA est plus performant dans les configurations à 8 et 16 îles. Les îles témoins permettent de parcourir de plus grandes régions de l'espace objectif pour ainsi permettre au modèle de fournir des ensembles de solutions mieux diversifiées.

Finalement, cette première étape de l'expérimentation a permis de révéler les bonnes performances du modèle proposé APM-MOEA par rapport aux modèles de la littérature sur les bases d'une parallélisation de l'algorithme GISMOO. En se basant sur les quatre métriques multi-objectifs calculées, il apparaît que APM-MOEA obtient de meilleures performances. En effet, sa capacité à mieux converger vers le front Pareto approximé et à trouver une meilleure distribution des solutions a été ainsi mise en évidence. La section suivante aborde les performances du modèle APM-MOEA vis-à-vis d'un modèle parallèle récemment proposé.

5.2.5 COMPARAISON À PASMOQAP

PasMoQAP (Sanhueza *et al.*, 2017) est un algorithme parallèle spécialisé dans la résolution du problème de MQAP. Il est basé sur un paradigme en îles sur lesquelles s'exécute un algorithme mémétique (algorithme génétique et recherche locale). Dans les travaux originaux, les auteurs ont optimisé les instances de MQAP à 60 unités en utilisant 5, 8, 11, 16 et 21 cœurs de processeur et en fixant un temps d'exécution maximum de 300 secondes. Les mêmes configurations ont été utilisées pour APM-MOEA mais en accordant 200 secondes pour chaque instance en raison d'un environnement d'expérimentation différent. Les ensembles de solutions rendus disponibles par les auteurs ont été utilisés pour comparer directement les modèles.

Dans un premier temps, les résultats de la métrique de couverture $\mathscr{C}$ de deux ensembles sont présentés à la Figure 5.15. Elle regroupe les valeurs moyennes de $\mathscr{C}$(APM-MOEA,PasMoQAP) et $\mathscr{C}$(PasMoQAP,APM-MOEA) pour chacune des configurations.

Les résultats obtenus sont amplement en faveur du modèle proposé puisque sur toutes les instances de MQAP les mesures de couverture $\mathscr{C}$(APM-MOEA,PasMoQAP) sont égales à $1,00$ et les valeurs de $\mathscr{C}$(PasMoQAP,APM-MOEA) à $0,00$. Autrement dit, toutes les solutions fournies par APM-MOEA ne sont pas dominées par celles de PasMoQAP. Au contraire, les solutions de PasMoQAP sont dominées par au moins une solution trouvée par APM-MOEA. Les résultats de cette métrique suffisent à montrer que le modèle proposé à de manière

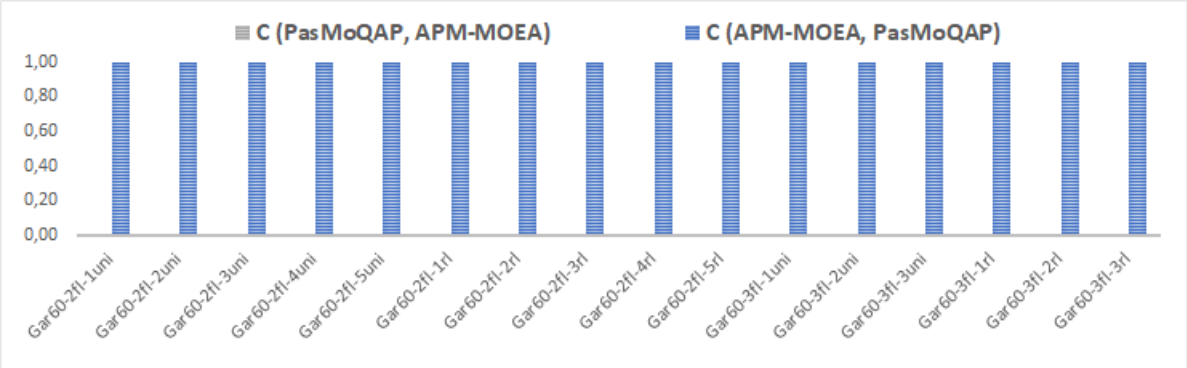


Figure 5.15: Couverture $\mathscr{C}$ moyenne de APM-MOEA par rapport à PasMoQAP sur les instances de MQAP à 60 unités

significative une meilleure capacité à converger vers le front Pareto approximé pour chacune des configurations. La recherche locale, qui présente un même niveau d'intensification dans les deux modèles, ne permet pas d'atteindre la même qualité de solution. La répartition des îles dans l'espace objectif proposée par APM-MOEA et les opérations évolutionnaires de GISMOO permettent d'atteindre une capacité d'intensification plus importante.

En ce qui concerne la diversité des solutions, la métrique d'espacement minimal *ms* a été calculée pour chaque instance et les valeurs moyennes sont exposées au Tableau 5.12.

	5 îles	8 îles	11 îles	16 îles	21 îles
PasMoQAP	0,0715	0,0690	0,0876	0,0881	0,0624
APM-MOEA	**0,0151**	**0,0134**	**0,0127**	**0,0124**	**0,0123**

Tableau 5.12: Espacement minimal *ms* moyen de PasMoQAP et APM-MOEA sur les instances de MQAP à 60 unités

D'après les résultats de la mesure d'espacement minimal *ms*, il apparaît que le modèle APM-MOEA trouve, en moyenne, une meilleure répartition des solutions que l'algorithme comparé. En effet, les valeurs *ms* sont largement inférieures pour le modèle proposé dans toutes les configurations. Par ailleurs, contrairement à APM-MOEA, l'algorithme PasMoQAP présente des difficultés à améliorer la distribution de ses solutions avec l'augmentation du nombre d'îles. Les deux métriques présentées permettent de constater de la supériorité du modèle proposé APM-MOEA vis-à-vis de l'algorithme parallèle PasMoQAP que ce soit en termes de convergence vers le front Pareto ou de diversité des solutions obtenues. Afin de vérifier graphiquement les résultats numériques, des exemples de solutions obtenues lors d'une

exécution typique sont présentés aux Figures 5.16, 5.17, 5.18 et 5.19 pour des instances à deux ou trois objectifs.

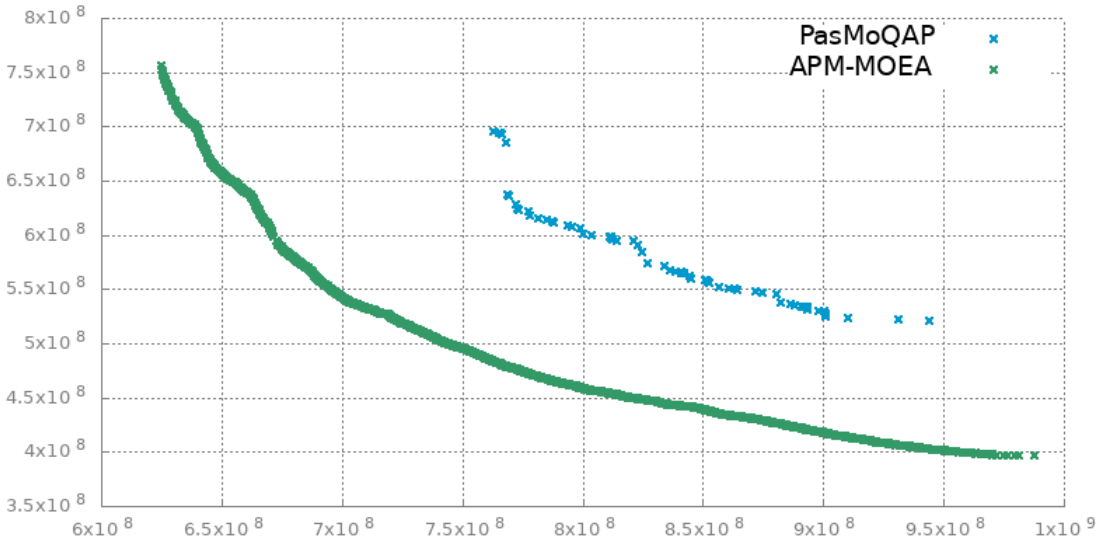


Figure 5.16: Représentation graphique pour la résolution typique de l'instance *Gar60-2fl-1rl* par les modèles APM-MOEA et PasMoQAP en utilisant 8 îles

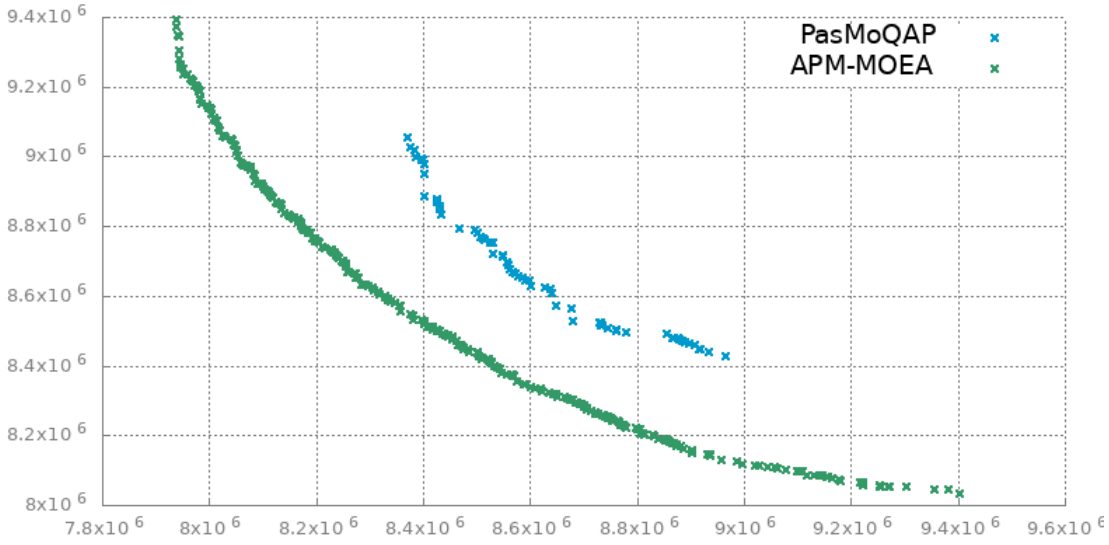


Figure 5.17: Représentation graphique pour la résolution typique de l'instance *Gar60-2fl-1uni* par les modèles APM-MOEA et PasMoQAP en utilisant 16 îles

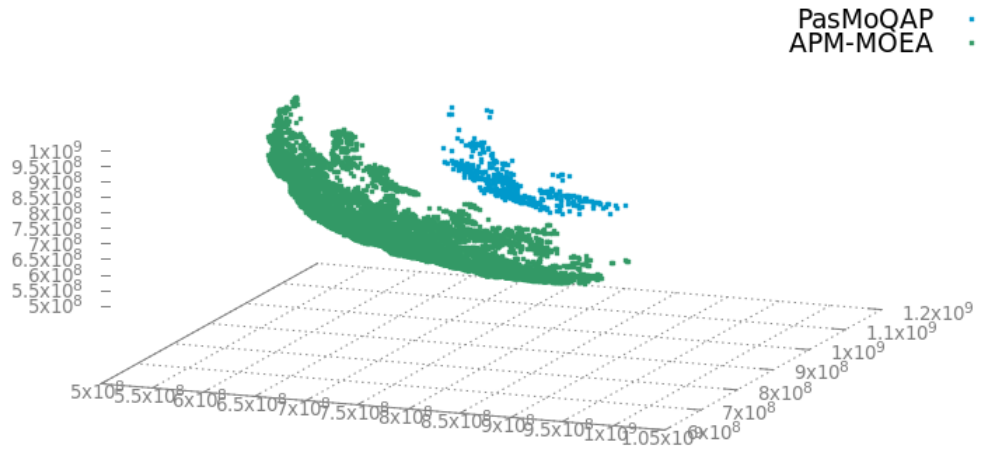


Figure 5.18: Représentation graphique pour la résolution typique de l'instance *Gar60-3fl-1rl* par les modèles APM-MOEA et PasMoQAP en utilisant 8 îles

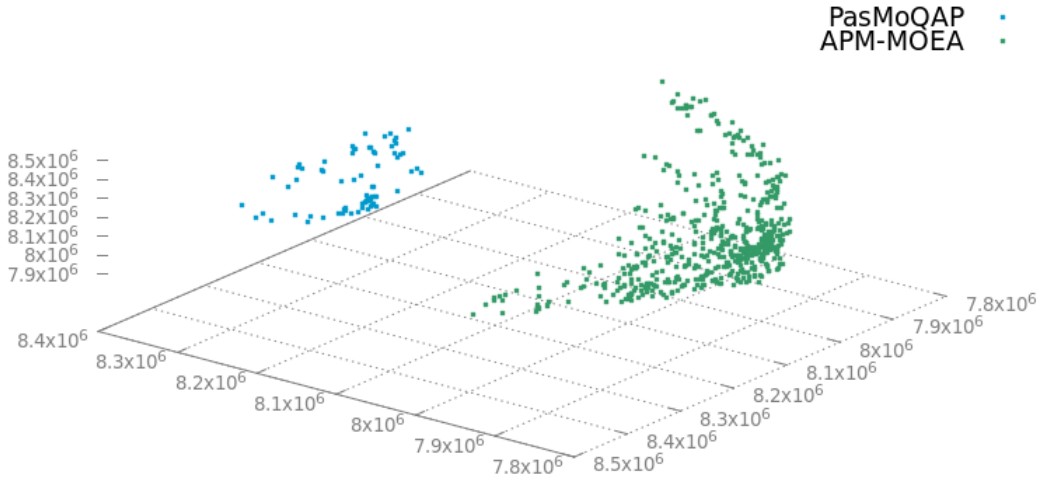


Figure 5.19: Représentation graphique pour la résolution typique de l'instance *Gar60-3fl-3uni* par les modèles APM-MOEA et PasMoQAP en utilisant 16 îles

Au vu des exemples proposés, il apparaît que le modèle APM-MOEA obtient des solutions de meilleure qualité que l'algorithme PasMoQAP. Des distances importantes sont observées que ce soit pour les instances à deux objectifs ou les instances à trois objectifs. La bonne distribution des solutions de APM-MOEA est également observée à travers ces espaces objectif. La supériorité du modèle proposé est ainsi confirmée de manière graphique et valide

les résultats obtenus à travers les différentes métriques.

Au final, les deux premières parties de l'expérimentation ont permis de mettre en évidence les performances du modèle sur des instances d'une taille relativement faible (60 variables de décision). Les métriques calculées ont montré que APM-MOEA surclasse des modèles de la littérature dans la parallélisation de l'algorithme GISMOO et qu'il est très compétitif vis-à-vis d'un algorithme parallèle récemment proposé. La prochaine section s'intéresse à la résolution d'instances de plus grande taille.

5.2.6 RÉSULTATS SUR DES INSTANCES DE GRANDE TAILLE

Dans la littérature concernant la résolution du problème de MQAP, il n'existe pas à proprement parler d'instances de test de grande taille. Pour vérifier les performances du modèle proposé avec de telles instances, le générateur proposé par Knowles et Corne (2003) a été utilisé. Ainsi, 22 instances de 100 à 1000 variables de décision ont été créées [1] et leurs caractéristiques sont exposées au Tableau 5.13 : le nom de l'instance (*Nom*), le nombre de sites/unités (*Taille*), le type de générateur utilisé pour la création (uniforme ou réel), le nombre d'objectifs (*Obj.*) et le paramètre de corrélation entre le premier et les autres objectifs (Corr.).

Nom	Taille	Type	Obj.	Corr.
Gar100-2fl-1uni	100	Uni	2	-0,5
Gar100-2fl-2uni	100	Uni	2	0
Gar100-2fl-3uni	100	Uni	2	0,5
Gar100-2fl-1rl	100	Réél	2	-0,5
Gar100-2fl-2rl	100	Réél	2	0
Gar100-2fl-3rl	100	Réél	2	0,5
Gar200-2fl-1uni	200	Uni	2	0
Gar200-2fl-2uni	200	Uni	2	0,5
Gar200-2fl-1rl	200	Réél	2	0
Gar200-2fl-2rl	200	Réél	2	0,5
Gar300-2fl-1uni	300	Uni	2	0

Nom	Taille	Type	Obj	Corr.
Gar300-2fl-2uni	300	Uni	2	0,5
Gar300-2fl-1rl	300	Réél	2	0
Gar300-2fl-2rl	300	Réél	2	0,5
Gar500-2fl-1uni	500	Uni	2	0
Gar500-2fl-2uni	500	Uni	2	0,5
Gar500-2fl-1rl	500	Réél	2	0
Gar500-2fl-2rl	500	Réél	2	0,5
Gar1000-2fl-1uni	1000	Uni	2	0
Gar1000-2fl-2uni	1000	Uni	2	0,5
Gar1000-2fl-1rl	1000	Réél	2	0
Gar1000-2fl-2rl	1000	Réél	2	0,5

Tableau 5.13: Instances de MQAP de 100 à 1000 variables de décision générées par le générateur de Knowles et Corne (2003)

1. Les instances de MQAP générées sont disponibles à l'adresse `https://sourceforge.net/projects/apm-moea-mqap-instances/`. Les solutions non-dominées trouvées par APM-MOEA sur chacune des instances sont également accessibles.

Comme le temps d'évaluation d'une solution de MQAP augmente considérablement avec le nombre de variables de décision, différents temps ont été alloués pour chaque type d'instance. Par conséquent, les temps d'exécution maximum pour les instances à 100, 200, 300, 500 et 1000 unités sont respectivement de 300, 500, 1000, 1500 et 9000 secondes. Les fronts Pareto ont été approximés à travers les ensembles obtenus de tous les modèles et dans toutes les configurations. Dans la première partie de ces expériences, la capacité de convergence de APM-MOEA et des autres modèles de la littérature est analysée à travers la métrique de convergence *IGD* dans laquelle les fonctions objectif sont normalisées entre 0 et 1. Les résultats moyens de toutes les instances sont présentés aux Tableaux 5.14, 5.15, 5.16 et 5.17 pour respectivement les configurations à 2, 4, 8 et 16 îles.

2 îles	DRMOGA	Séparation en cônes	Clustering original	APM-MOEA
Gar100-2fl-1uni	0,0411	0,0436	**0,0365**	0,0373
Gar100-2fl-2uni	0,0561	0,0929	0,0580	**0,0475**
Gar100-2fl-3uni	0,1817	**0,1013**	0,1118	0,1949
Gar100-2fl-1rl	0,0537	0,0681	0,0427	**0,0375**
Gar100-2fl-2rl	0,0437	0,0763	**0,0294**	0,0520
Gar100-2fl-3rl	0,0830	0,0780	0,0691	**0,0681**
Gar200-2fl-1uni	0,1504	0,1599	0,1657	**0,1470**
Gar200-2fl-2uni	0,2313	**0,1920**	0,1957	0,2819
Gar200-2fl-1rl	0,0595	0,0607	**0,0353**	0,0558
Gar200-2fl-2rl	0,0603	0,0580	**0,0410**	0,0431
Gar300-2fl-1uni	0,1950	0,2210	0,2079	**0,1838**
Gar300-2fl-2uni	**0,4107**	0,4495	0,4534	0,4127
Gar300-2fl-1rl	0,1106	0,1805	0,1724	**0,1082**
Gar300-2fl-2rl	**0,1143**	0,1850	0,1234	0,1951
Gar500-2fl-1uni	0,2360	**0,2040**	0,2479	0,2317
Gar500-2fl-2uni	0,2277	0,2706	0,2968	**0,1844**
Gar500-2fl-1rl	**0,1795**	0,1872	0,3224	0,2087
Gar500-2fl-2rl	**0,1715**	0,2567	0,2724	0,2337
Gar1000-2fl-1uni	0,2739	0,3206	0,3192	**0,2495**
Gar1000-2fl-2uni	**0,5220**	0,7176	0,7436	0,6097
Gar1000-2fl-1rl	0,2997	0,3902	0,3899	**0,2930**
Gar1000-2fl-2rl	**0,3254**	0,4149	0,4364	0,3813

Tableau 5.14: Convergence *IGD* moyenne de de DRMOGA, Séparation en cônes, Clustering original et APM-MOEA sur les instances de MQAP de grande taille en utilisant 2 îles

4 îles	DRMOGA	Séparation en cônes	Clustering original	APM-MOEA
Gar100-2fl-1uni	**0,0213**	0,0310	0,0270	0,0388
Gar100-2fl-2uni	0,0418	**0,0378**	0,0528	0,0561
Gar100-2fl-3uni	0,1896	0,2125	**0,0279**	0,0341
Gar100-2fl-1rl	0,0425	0,0263	0,0271	**0,0243**
Gar100-2fl-2rl	0,0320	0,0461	0,0383	**0,0204**
Gar100-2fl-3rl	0,0574	0,0819	0,0470	**0,0435**
Gar200-2fl-1uni	0,1053	0,1058	0,1059	**0,0894**
Gar200-2fl-2uni	0,2166	0,1942	0,0967	**0,0578**
Gar200-2fl-1rl	0,0435	0,0597	**0,0349**	0,0525
Gar200-2fl-2rl	0,0562	0,0883	0,0490	**0,0405**
Gar300-2fl-1uni	0,1292	0,1278	**0,1231**	0,1314
Gar300-2fl-2uni	0,3240	0,3851	0,4022	**0,2401**
Gar300-2fl-1rl	0,0721	0,1319	0,2927	**0,0154**
Gar300-2fl-2rl	0,1213	0,1737	0,3751	**0,0719**
Gar500-2fl-1uni	0,1493	0,1704	0,1991	**0,1472**
Gar500-2fl-2uni	0,1767	0,2171	0,2266	**0,0759**
Gar500-2fl-1rl	0,1464	0,1818	0,3423	**0,0691**
Gar500-2fl-2rl	0,1837	0,3054	0,4199	**0,0431**
Gar1000-2fl-1uni	0,2031	0,2150	0,2322	**0,1378**
Gar1000-2fl-2uni	0,4895	0,5198	0,6490	**0,3286**
Gar1000-2fl-1rl	0,2447	0,3492	0,3913	**0,1340**
Gar1000-2fl-2rl	0,2577	0,4560	0,4509	**0,1303**

Tableau 5.15: Convergence *IGD* moyenne de de DRMOGA, Séparation en cônes, Clustering original et APM-MOEA sur les instances de MQAP de grande taille en utilisant 4 îles

8 îles	DRMOGA	Séparation en cônes	Clustering original	APM-MOEA
Gar100-2fl-1uni	0,0206	0,0236	**0,0090**	0,0270
Gar100-2fl-2uni	0,0332	0,0380	0,0276	**0,0187**
Gar100-2fl-3uni	0,1117	0,1302	0,0836	**0,0835**
Gar100-2fl-1rl	0,0166	0,0313	**0,0108**	0,0264
Gar100-2fl-2rl	0,0206	0,0447	0,0246	**0,008**
Gar100-2fl-3rl	0,0421	0,0736	0,0512	**0,0324**
Gar200-2fl-1uni	0,0627	0,0767	**0,0541**	0,0728
Gar200-2fl-2uni	0,1059	0,1437	0,2164	**0,1057**
Gar200-2fl-1rl	0,0455	0,0541	**0,0346**	0,0428
Gar200-2fl-2rl	**0,0252**	0,0392	0,0337	0,0265
Gar300-2fl-1uni	0,0945	0,0609	**0,0346**	0,0965
Gar300-2fl-2uni	0,3265	0,3035	0,2691	**0,2475**
Gar300-2fl-1rl	**0,0598**	0,1332	0,3064	0,0776
Gar300-2fl-2rl	0,0406	0,1842	0,4987	**0,0273**
Gar500-2fl-1uni	0,1332	0,1118	0,0892	**0,0619**
Gar500-2fl-2uni	0,1347	0,1948	0,1552	**0,0447**
Gar500-2fl-1rl	0,0642	0,2342	0,3834	**0,0165**
Gar500-2fl-2rl	0,1015	0,2324	0,4657	**0,0699**
Gar1000-2fl-1uni	0,1552	0,1434	0,0952	**0,0272**
Gar1000-2fl-2uni	0,4403	0,3495	0,4777	**0,2699**
Gar1000-2fl-1rl	0,1771	0,3236	0,3500	**0,0051**
Gar1000-2fl-2rl	0,1425	0,4171	0,4374	**0,0588**

Tableau 5.16: Convergence *IGD* moyenne de de DRMOGA, Séparation en cônes, Clustering original et APM-MOEA sur les instances de MQAP de grande taille en utilisant 8 îles

16 îles	DRMOGA	Séparation en cônes	Clustering original	APM-MOEA
Gar100-2fl-1uni	0,0088	0,0152	**0,0051**	0,0147
Gar100-2fl-2uni	0,0177	0,0337	0,0223	**0,0166**
Gar100-2fl-3uni	0,0179	0,1405	0,0491	**0,0122**
Gar100-2fl-1rl	0,0243	0,0219	0,0159	**0,0147**
Gar100-2fl-2rl	0,0373	0,0240	0,0227	**0,0117**
Gar100-2fl-3rl	0,0270	0,0508	0,0291	**0,0198**
Gar200-2fl-1uni	0,0456	0,0339	**0,0023**	0,0607
Gar200-2fl-2uni	0,1402	0,1018	0,1029	**0,0578**
Gar200-2fl-1rl	0,0097	0,0463	0,0174	**0,0044**
Gar200-2fl-2rl	0,0218	0,0356	0,0204	**0,0088**
Gar300-2fl-1uni	0,0507	0,0288	0,0608	**0,0062**
Gar300-2fl-2uni	0,2181	0,1772	0,2520	**0,1276**
Gar300-2fl-1rl	**0,0416**	0,1353	0,3461	0,0440
Gar300-2fl-2rl	0,0993	0,1363	0,3880	**0,0094**
Gar500-2fl-1uni	0,0445	0,0535	0,0240	**0,0155**
Gar500-2fl-2uni	0,1098	0,1704	0,0916	**0,01136**
Gar500-2fl-1rl	0,0615	0,2350	0,3745	**0,0027**
Gar500-2fl-2rl	0,1149	0,2639	0,5208	**0,0041**
Gar1000-2fl-1uni	0,0662	0,1072	0,0459	**0,0126**
Gar1000-2fl-2uni	0,3254	0,4202	0,4162	**0,0803**
Gar1000-2fl-1rl	0,1342	0,3214	0,3956	**0,015**
Gar1000-2fl-2rl	0,1866	0,4434	0,4415	**0,0021**

Tableau 5.17: Convergence *IGD* moyenne de de DRMOGA, Séparation en cônes, Clustering original et APM-MOEA sur les instances de MQAP de grande taille en utilisant 16 îles

En considérant la métrique de convergence *IGD*, ce sont les modèles DRMOGA et APM-MOEA qui montrent les meilleures performances en utilisant seulement 2 îles. Le premier obtient les plus faibles moyennes de convergence *IGD* pour 6 des 22 instances alors que le second propose une meilleure convergence pour 9 des 22 instances. Dans les configurations à 4 et 8 îles, APM-MOEA fournit des solutions plus proches du front Pareto pour la majorité des instances, particulièrement pour les instances de plus grande taille, c'est-à-dire avec 500 ou 1000 unités. En utilisant 16 îles, APM-MOEA propose une qualité de solution supérieure aux autres modèles puisqu'il obtient les plus faibles valeurs de convergence *IGD* pour 19 des 22 instances. De plus, l'extensibilité du modèle est mise en évidence puisqu'une meilleure qualité de solution est obtenue en augmentant le nombre d'îles.

Dans le but de confirmer la capacité de convergence de APM-MOEA et d'analyser les performances pour chaque instance de MQAP, la mesure de couverture $\mathscr{C}$ de deux ensembles a été calculée et les résultats pour 16 îles sont exposés à la Figure 5.20.

Figure 5.20: Couverture $\mathscr{C}$ moyenne de APM-MOEA par rapport à DRMOGA, Séparation en cônes et Clustering original en utilisant 16 îles sur les instances de MQAP de grande taille

Pour la majorité des instances, les valeurs moyennes de couverture $\mathscr{C}$(APM-MOEA,DRMOGA), $\mathscr{C}$(APM-MOEA, Séparation en cônes) et $\mathscr{C}$(APM-MOEA,Clustering original) sont très élevées et sont comprises entre $0,7$ et $1,00$. Elles indiquent une meilleure convergence de APM-MOEA puisqu'il trouve une majorité de solutions qui dominent celles des autres modèles. En outre, les valeurs proches de 0 pour les mesures $\mathscr{C}$(DRMOGA, APM-MOEA), $\mathscr{C}$(Séparation en cônes, APM-MOEA) et $\mathscr{C}$(Clustering original, APM-MOEA) indiquent que les modèles de la littérature trouvent des solutions qui ne dominent pas celle de APM-MOEA. Dans la résolution des plus grandes instances, avec plus de 500 variables de décision, le modèle

proposé s'avère encore plus performant puisqu'il obtient des couvertures proches de $1,00$ sur les autres modèles. La capacité de convergence du modèle est donc justifiée par les deux premières métriques et sa supériorité est encore plus importante sur les instances de plus grandes tailles.

Pour confirmer les bonnes performances du modèle `APM-MOEA`, la métrique d'hypervolume $\mathscr{S}$ a été calculée et les valeurs moyennes de toutes les instances sont exposées à la Figure 5.21.

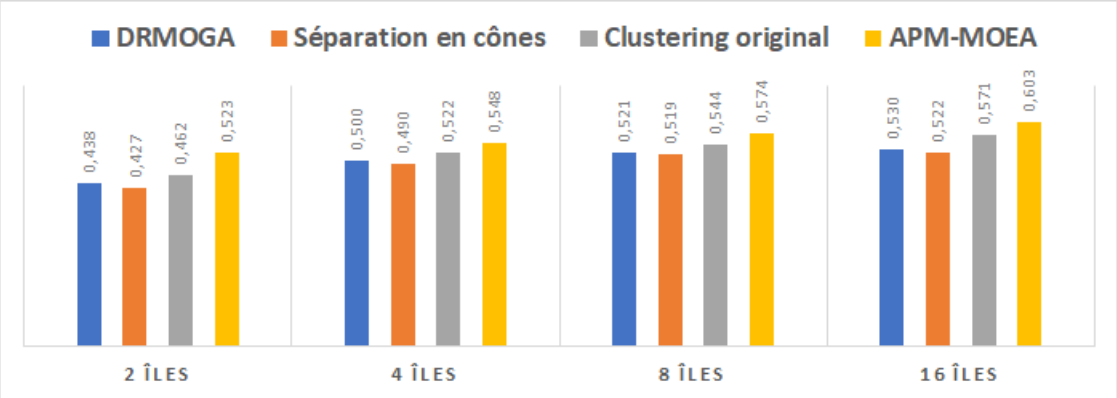


Figure 5.21: Hypervolume $\mathscr{S}$ moyen de `DRMOGA`, `Séparation en cônes`, `Clustering original` et `APM-MOEA` sur les grandes instances de MQAP

Dans toutes les configurations, le modèle `APM-MOEA` obtient une meilleure mesure d'hypervolume $\mathscr{S}$. Les autres modèles ont plus de difficultés à recouvrir l'espace objectif, notamment `DRMOGA` qui proposait pourtant une bonne convergence selon la métrique de convergence *IGD*. Les îles témoins dans `APM-MOEA` permettent de couvrir une partie plus importante de l'espace objectif comme l'indiquent les plus grandes valeurs de couverture $\mathscr{C}$ obtenues avec l'utilisation de 8 et 16 îles. Tout comme la métrique de convergence *IGD* le suggérait, de meilleures performances sont atteintes dans les plus grandes configurations. Finalement, la répartition des solutions est analysée à travers la mesure d'espacement minimal *ms* et les valeurs moyennes obtenues par les modèles parallèles sont exposées au Tableau 5.18.

Dans la plus petite des configurations, le modèle `DRMOGA` ne parvient pas cette fois à obtenir des résultats comparables à `APM-MOEA`. Son habilité à trouver des solutions diversifiées n'est donc pas affirmée par cette métrique. En revanche, dans toutes les configurations, le modèle `APM-MOEA` obtient les plus petites valeurs d'espacement minimal *ms*. La diversité des solutions est améliorée en augmentant le nombre d'îles comme l'indique la diminution de la valeur

d'espacement minimal *ms*. Cette métrique montre que APM-MOEA fournit des solutions mieux réparties dans l'espace objectif que les autres modèles.

	2 îles	4 îles	8 îles	16 îles
DRMOGA	0,539	0,527	0,517	0,504
Séparation en cônes	0,507	0,512	0,520	0,528
Clustering original	0,519	0,516	0,503	0,497
APM-MOEA	**0,497**	**0,484**	**0,467**	**0,442**

Tableau 5.18: Espacement minimal moyen de DRMOGA, Séparation en cônes, Clustering original et APM-MOEA sur les grandes instances de MQAP

Finalement, afin d'attester des performances des différents modèles parallèles implémentés, des exemples de solutions obtenues pour des exécutions typiques sont proposés aux Figures 5.22 et 5.23. La capacité de convergence du modèle APM-MOEA est notamment mise en avant à travers les exemples puisque celui-ci obtient les solutions de meilleure qualité confirmant ainsi les tendances indiquées par les métriques multi-objectifs présentées.

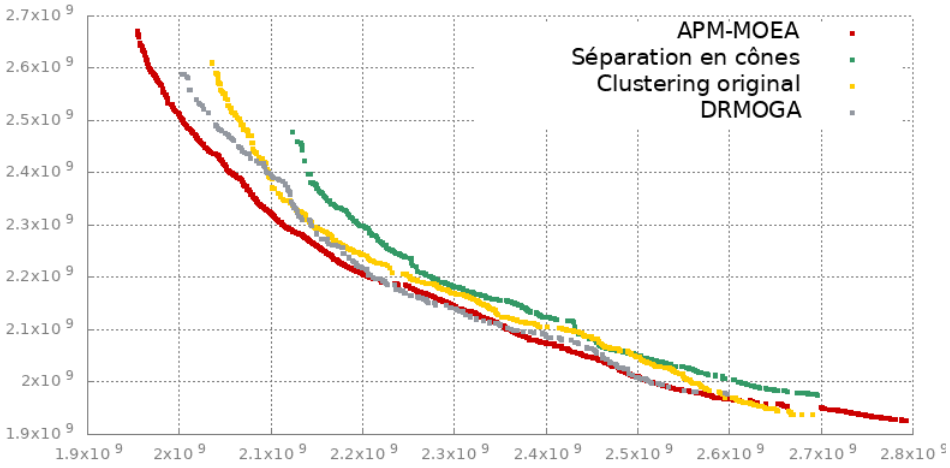


Figure 5.22: Représentation graphique pour la résolution typique de l'instance *Gar200-2fl-1rl* par APM-MOEA et les trois modèles parallèles de la littérature en utilisant 8 îles

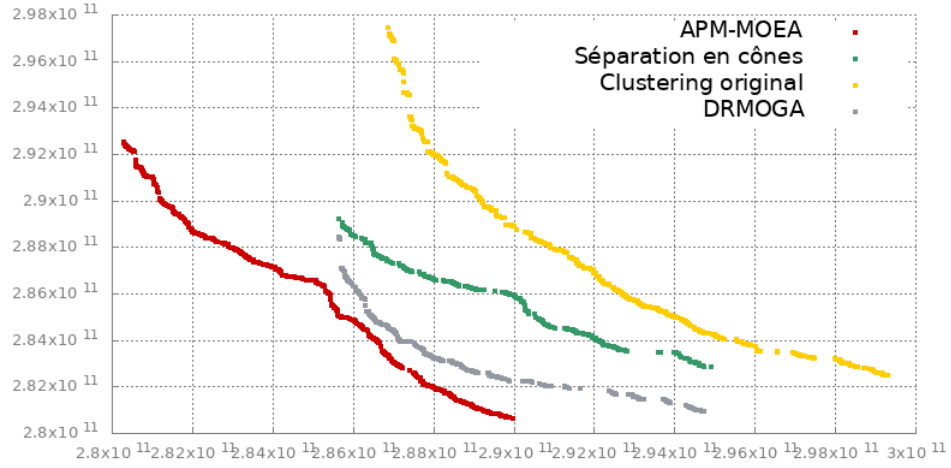


Figure 5.23: Représentation graphique pour la résolution typique de l'instance *Gar1000-2fl-1rl* par APM-MOEA et les trois modèles parallèles de la littérature en utilisant 16 îles

5.2.7 CONCLUSION SUR LA RÉSOLUTION DE MQAP

La résolution des instances de MQAP par le biais du modèle parallèle APM-MOEA a mis en exergue les qualités du modèle proposé. Que ce soit sur des instances avec peu de variables de décision ou des instances générées de grande taille, le modèle APM-MOEA s'est révélé plus performant que les modèles parallèles de la littérature. Sa capacité à converger vers le front Pareto et à explorer de manière efficace l'espace objectif a été également mise en avant par sa comparaison avantageuse avec un algorithme parallèle spécialisé dans la résolution du MQAP, c'est-à-dire l'algorithme PasMoQAP (Sanhueza *et al.*, 2017). Finalement, il est important de noter que le comportement de APM-MOEA semble encore plus intéressant lors de la résolution d'instances de grande taille comme l'ont montré les résultats sur les instances à 500 et 1000 variables de décision.

5.3 CONCLUSION

Ce chapitre a permis de mettre en évidence les performances du modèle parallèle APM-MOEA dans la résolution de deux problèmes multi-objectifs classiques. Sur des instances de MOTSP et de MQAP, APM-MOEA parvient à fournir des solutions de meilleure qualité que des modèles parallèles proposés dans la littérature. Sa capacité à mieux converger vers le front Pareto a été révélée par différentes métriques. De même, le modèle parvient à fournir des ensembles de solutions diversifiés, notamment grâce aux îles témoins qui permettent

d'explorer de manière efficace l'espace objectif. Son extensibilité a également été démontrée à travers les différentes métriques puisqu'il arrive à améliorer la qualité de ses solutions avec l'augmentation du nombre de cœurs de processeur.

Le modèle APM-MOEA s'est également révélé compétitif vis-à-vis d'algorithmes de résolution spécifiques tels que 2PPLS (Lust et Teghem, 2010c) et PasMoQAP (Sanhueza *et al.*, 2017). Même s'il n'arrive pas à atteindre les performances du premier avec la parallélisation de GISMOO, il permet à l'algorithme d'obtenir des solutions approchées. En revanche, il apparaît que le modèle APM-MOEA surclasse significativement l'algorithme parallèle PasMoQAP puisque sur toutes les instances de MQAP, APM-MOEA obtient une meilleure qualité de solution.

CONCLUSION ET PERSPECTIVES

Dans de nombreux secteurs, les problèmes rencontrés sont complexes et impliquent l'optimisation de plusieurs objectifs conflictuels. Pour répondre à de tels problèmes, l'utilisation de méthodes approchées, comme les métaheuristiques, est nécessaire afin de proposer des solutions en un temps qualifié d'acceptable. Les travaux de recherche effectués dans le cadre de cette thèse ont permis de mettre en évidence l'intérêt des métaheuristiques et du parallélisme dans la résolution de problèmes multi-objectifs. En plus d'accélérer l'exécution d'un algorithme évolutionnaire, l'exploitation de plusieurs unités de calcul peut permettre d'améliorer les performances de l'algorithme en répartissant généralement la recherche du front Pareto sur plusieurs processus. En suivant l'objectif général de thèse qui consistait à *contribuer à amener les modèles parallèles conçus pour les métaheuristiques multi-objectifs à exploiter efficacement les architectures de calcul haute performance*, de nouvelles stratégies de parallélisme ont été proposées au sein d'un modèle parallèle original.

Afin de parvenir au premier objectif qui visait à *proposer un nouveau modèle parallèle adapté aux algorithmes évolutionnaires multi-objectifs*, il a été nécessaire de comprendre les enjeux et les difficultés des deux grands domaines concernés par ces travaux de recherche, c'est-à-dire l'optimisation multi-objectifs et la programmation parallèle. Pour ce faire, les principaux algorithmes évolutionnaires multi-objectifs et plusieurs modèles parallèles ont été présentés et étudiés dans les deux premiers chapitres. L'identification des avantages et des limites des différents modèles parallèles a constitué les prémices dans la proposition d'un nouveau modèle parallèle correspondant au premier objectif spécifique de la thèse. Plus précisément, le troisième chapitre a permis de présenter le modèle APM-MOEA qui est basé sur le paradigme en îles et utilise un algorithme de partitionnement original pour répartir la

recherche du front Pareto sur les différentes îles. Les principales caractéristiques et apports de APM-MOEA sont :

- Chaque île est affectée à une partie spécifique de l'espace objectif selon les principes de dominance sous contrainte ;
- Une vue globale est donnée à l'île organisatrice à travers le maintien d'une archive de solutions non-dominées ;
- Des échanges de communications asynchrones entre les îles, notamment pour l'échange d'archives locales avec l'île organisatrice pour limiter les surcoûts du modèle ;
- Des îles additionnelles, nommées îles témoins, pour améliorer l'exploration et fournir des solutions diversifiées ;
- Une amélioration locale qui est appliquée périodiquement pour améliorer la capacité de converger de l'algorithme.

En suivant le deuxième objectif spécifique de cette thèse qui consistait à *établir une comparaison expérimentale des principaux modèles parallèles adaptés aux algorithmes évolutionnaires multi-objectifs*, les implémentations parallèles de l'algorithme GISMOO (Zinflou *et al.*, 2012) avec APM-MOEA et plusieurs modèles de la littérature ont été confrontées. L'atteinte de cet objectif a permis de mettre en évidence les avantages et les limites des modèles existants. Les modèles de séparation de l'espace objectif sont par exemple performants en exploitant quelques cœurs de processeur et dans la résolution d'instances présentant des fronts Pareto très fournis en solutions. Cependant, ils ne parviennent pas à améliorer leur qualité de solution avec un nombre important d'îles puisque les îles sont affectées à des portions de l'espace objectif trop réduites. Les autres modèles sont plus adaptés pour approximer des fronts Pareto pauvres en solutions et proposent une meilleure diversité avec un nombre élevé d'îles. Par ailleurs, l'apport de certaines composantes spécifiques de APM-MOEA a été analysé. Les échanges de communications asynchrones réduisent d'une part les surcoûts du modèle parallèle comme l'a révélé le profilage de code effectué. D'autre part, la vue globale de l'île organisatrice et le nouvel algorithme de partitionnement améliorent la capacité d'exploration du modèle et permettent de fournir des solutions mieux distribuées dans l'espace objectif.

Le dernier objectif consistait à *résoudre efficacement des problèmes combinatoires multi-objectifs à travers le modèle proposé et analyser les performances*. Cet objectif a été complété lors du Chapitre 5 grâce aux versions multi-objectifs du problème de voyageur de commerce et du problème d'affectation quadratique. Lors de la résolution du MOTSP, l'apport des îles

témoins dans la capacité d'exploration du modèle a été exposé. De même, il a été montré qu'il est presque indispensable d'avoir une forme de recherche locale pour obtenir une méthode de résolution compétitive par rapport à celles de la littérature. Sur plusieurs instances de MOTSP et de MQAP, le modèle s'est révélé très performant par rapport à des modèles de séparation de l'espace objectif en termes de convergence vers le front Pareto et en termes de distribution des solutions. La capacité du modèle à améliorer la qualité des solutions avec l'augmentation du nombre d'îles a été également montrée et dénote ainsi de sa bonne extensibilité. En ce qui concerne le MOTSP, la parallélisation de GISMOO à travers le modèle APM-MOEA ne permet pas d'atteindre les performances d'un algorithme spécialisé dans la résolution du MOTSP comme 2PPLS Lust et Teghem (2010c). Néanmoins, le modèle permet d'approcher les solutions d'un algorithme générique comme GISMOO des meilleures solutions trouvées dans la littérature. Pour les instances de MQAP, le modèle APM-MOEA surclasse un algorithme parallèle spécialisé dans la résolution du problème : l'algorithme PasMoQAP (Sanhueza *et al.*, 2017). Sa capacité à mieux converger vers le front Pareto et à explorer de manière plus efficace l'espace objectif a été en effet mise en évidence.

En résumé, les travaux effectués pour les trois objectifs spécifiques ont permis d'atteindre l'objectif général de cette thèse qui était de *contribuer à amener les modèles parallèles conçus pour les métaheuristiques multi-objectifs à exploiter efficacement les architectures de calcul haute performance*. Le modèle APM-MOEA permet aux méthodes de résolution d'exploiter efficacement plusieurs unités de calcul. Ainsi, à travers la parallélisation d'un algorithme multi-objectifs générique, une méthode de résolution compétitive par rapport aux algorithmes de la littérature a été obtenue. Les temps d'exécution sont, par ailleurs, diminués grâce aux communications asynchrones. Finalement, ces recherches ont montré l'apport significatif que peut apporter le parallélisme aux algorithmes multi-objectifs.

Bien que le modèle proposé permette de paralléliser différents algorithmes évolutionnaires multi-objectifs, cette thèse s'est limitée à la parallélisation de l'algorithme GISMOO. La généricité du modèle reste donc à étudier dans de futures recherches qui permettraient d'évaluer la capacité de APM-MOEA à améliorer la qualité des solutions fournies par d'autres algorithmes multi-objectifs. Comme le modèle utilise une stratégie *diviser pour régner*, les algorithmes basés sur la décomposition comme MOEA/D (Qingfu et Hui, 2007) ne pourront pas être parallélisés par ce modèle.

De même, le modèle APM-MOEA a fourni des résultats très prometteurs sur deux problèmes combinatoires spécifiques, mais son comportement permet la manipulation de différents types de problèmes. Il serait donc intéressant d'expérimenter le modèle APM-MOEA sur des problèmes à variables continues ou sur des problèmes à plus de trois objectifs qui n'ont pas été traités dans cette thèse. Pour obtenir de meilleurs résultats, il pourrait être nécessaire de paralléliser des algorithmes spécialisés dans la résolution du problème à optimiser.

Par ailleurs, les nouvelles architectures de calcul haute performance regroupent généralement plusieurs types d'architecture. Afin de profiter de leur puissance de calcul, l'hybridation de plusieurs modèles sera la solution privilégiée. Des travaux préliminaires concernant l'hybridation des modèles en îles et des modèles maître-esclave ont été effectués par Cantu-Paz (2000) et Jagtap *et al.* (2011). Ces travaux mettent en évidence autant le potentiel de l'hybridation que l'envergure des avancées pouvant encore être accomplies. Les deux phases qui concentrent la majorité des calculs (Veldhuizen *et al.*, 2003), c'est-à-dire les évaluations de fonctions objectif et les assignations de facteurs de dominance et d'isolement, seront parallélisées à travers le modèle maître-esclave. L'approche hybride sera alors un modèle parallèle à deux niveaux. Au haut niveau s'exécutera le modèle en îles APM-MOEA tel que décrit dans le Chapitre 3 et au bas niveau un modèle maître-esclave permettra d'accélérer les recherches.

De plus, comme les architectures parallèles sont en constante évolution, de futurs travaux seront consacrés à l'exportation du modèle sur de nouvelles architectures. Dans la résolution de problèmes industriels d'envergure, il sera nécessaire d'exploiter au mieux les nouveaux supercalculateurs et leur puissance de calcul. En outre, pour répondre à des problématiques de temps réel, l'utilisation d'architectures alternatives comme les GPUs ou les processeurs contenant plus de 500 cœurs (e.g. Intel Xeon Phi) sera envisagée dans de futurs travaux.

Finalement, les études expérimentales effectuées lors de cette thèse ont mis en avant la difficulté de comparer des modèles parallèles adaptés aux métaheuristiques multi-objectifs. D'une part, il est difficile de confronter des approches de résolution lorsque les fronts Pareto ne sont pas connus et disponibles. D'autre part, il n'existe pas de métrique spécifique pour évaluer les performances d'un modèle parallèle pour les MOEA. Il serait donc pertinent de proposer une démarche scientifique générique pour analyser l'apport d'un modèle parallèle dans l'amélioration de la qualité de solutions et dans l'accélération du temps d'exécution. Une nouvelle mesure pourrait alors être introduite pour estimer simultanément ces différents aspects tout en prenant en compte l'extensibilité de l'approche parallèle.

BIBLIOGRAPHIE

Adra, S., T. Dodd, I. Griffin, et P. Fleming. 2009. « Convergence acceleration operator for multiobjective optimization », *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 4, p. 825–847.

Alsheddy, A. et E. Tsang. 2010. « Guided pareto local search based frameworks for biobjective optimization ». In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, p. 1–8.

AMD. 2014. Ati stream, documentation en ligne, http ://docs.oracle.com/javase/7/docs/api/.

Angel, E., E. Bampis, et L. Gourvès. 2004. « A dynasearch neighborhood for the bicriteria traveling salesman problem ». In Gandibleux, X., M. Sevaux, K. Sörensen, et V. T'kindt, éditeurs, *Metaheuristics for Multiobjective Optimisation*, p. 153–176, Berlin, Heidelberg. Springer Berlin Heidelberg.

Angus, D. 2007. « Crowding population-based ant colony optimisation for the multi-objective travelling salesman problem ». In *Computational Intelligence in Multicriteria Decision Making, IEEE Symposium on*, p. 333–340.

Bader, J. et E. Zitzler. 2011. « Hype : An algorithm for fast hypervolume-based many-objective optimization », *Evolutionary Computation*, vol. 19, no. 1, p. 45–76.

Bandyopadhyay, S., S. Pal, et B. Aruna. 2004. « Multiobjective gas, quantitative indices, and pattern classification ». In *IEEE Trans Syst Man Cybern B Cybern*.

Bandyopadhyay, S., S. Saha, U. Maulik, et K. Deb. 2008. « A simulated annealing-based multiobjective optimization algorithm : Amosa », *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 3, p. 269–283.

Banos, R., C. Gil, B. Paechter, et J. Ortega. 2006. « Parallelization of Population-based Multi-objective Metaheuristics : An Empirical Study », *Applied Mathematical Modelling*, vol. 30, no. 7, p. 578–592.

Baran, B. et M. Schaerer. 2003. « A multiobjective ant colony system for vehicle routing problem with time windows », *The 21st IASTED International Multi-Conference on Applied, 2013*.

Berro, A. 2001. « Optimisation multiobjectif et stratégies d' évolution en environnement dynamique ». Thèse de Doctorat.

Beume, N., B. Naujoks, et M. Emmerich. 2007. « Sms-emoa : Multiobjective selection based on dominated hypervolume », *European Journal of Operational Research*, vol. 181, no. 3, p. 1653 – 1669.

Bleuler, S., M. Laumanns, L. Thiele, et E. Zitzler. 2003. « Pisa — a platform and programming language independent interface for search algorithms ». In Fonseca, C. M., P. J. Fleming, E. Zitzler, L. Thiele, et K. Deb, éditeurs, *Evolutionary Multi-Criterion Optimization*, p. 494–508, Berlin, Heidelberg. Springer Berlin Heidelberg.

Borges, P. C. et M. P. Hansen. 2002. *A Study of Global Convexity for a Multiple Objective Travelling Salesman Problem*, p. 129–150. Boston, MA : Springer US.

Branke, J., T. KauBler, et H. Schmeck. 2001. « Guidance in evolutionary multi-objective optimization ». In *Advances in Engineering Software*.

Branke, J., H. Schmeck, K. Deb, et M. Reddy S. 2004. « Parallelizing multi-objective evolutionary algorithms : cone separation ». In *Evolutionary Computation. CEC2004. Congress on*. T. 2, p. 1952–1957 Vol.2.

Brockhoff, D., T. Wagner, et H. Trautmann. 2015. « 2 indicator-based multiobjective search », *Evolutionary Computation*, vol. 23, no. 3, p. 369–395.

Brodtkorb, A. R., C. Dyken, T. R. Hagen, J. M. Hjelmervik, et O. O. Storaasli. 2010. « State-of-the-art in heterogeneous computing », *Scientific Programming*, vol. 18, no. 1, p. 1–33.

Bui, L. T., H. A. Abbass, et D. Essam. 2009. « Local models–an approach to distributed multi-objective optimization », *Computational Optimization and Applications*, vol. 42, no. 1, p. 105–139.

Bull. 2018. « Supercalculateurs bullsequana x1000 ». In *https ://atos.net/fr/produits/calcul-haute-performance-hpc/supercalculateurs-bull-sequana-x/gamme-bullsequana-x1000*.

Cahon, S., N. Melab, et E.-G. Talbi. 2004. « Paradiseo : A framework for the reusable design of parallel and distributed metaheuristics », *Journal of Heuristics*, vol. 10, no. 3, p. 357–380.

Cantu-Paz, E. 2000. *Efficient and Accurate Parallel Genetic Algorithms*. Norwell, MA, USA : Kluwer Academic Publishers.

Caponio, A. et F. Neri. 2009. *Integrating Cross-Dominance Adaptation in Multi-Objective Memetic Algorithms*. Coll. Goh, C.-K., Y.-S. Ong, et K. Tan, éditeurs, Coll. « *Multi-Objective Memetic Algorithms* ». T. 171, série *Studies in Computational Intelligence*, p. 325–351. Springer Berlin Heidelberg.

Cardoso, P., M. Jesus, et A. Marquez. 2003. *MONACO-multi-objective network optimisation based on an ACO*. Coll. « Proc. X Encuentros de Geometrıa Computacional, Seville, Spain, ».

Coello, C. 2001. *A Short Tutorial on Evolutionary Multiobjective Optimization*. Coll. Zitzler, E., L. Thiele, K. Deb, C. Coello Coello, et D. Corne, éditeurs, Coll. « *Evolutionary Multi-Criterion Optimization* ». T. 1993, série *Lecture Notes in Computer Science*, p. 21–40. Springer Berlin Heidelberg.

Coello, C. et G. Lamont. 2004. *Applications of Multi-objective Evolutionary Algorithms*. Coll. « Advances in natural computation ». World Scientific.

Coello, C. et M. Lechuga. 2002. « Mopso : A proposal for multiple objective particle swarm optimization », *Proceedings of the 2002 Congress on Evolutionary Computation part of the 2002 IEEE World Congress on Computational Intelligence*.

Coello, C. et M. Reyes Sierra. 2004. « A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm ». In Monroy, R., G. Arroyo-Figueroa, L. E. Sucar, et H. Sossa, éditeurs, *MICAI 2004 : Advances in Artificial Intelligence*, p. 688–697, Berlin, Heidelberg. Springer Berlin Heidelberg.

Corne, D. W., N. R. Jerram, J. D. Knowles, M. J. Oates, et M. J. 2001. « Pesa-ii : Region-based selection in evolutionary multiobjective optimization », *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*.

Czyżżak, P. et A. Jaszkiewicz. 1998. « Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization », *Journal of Multi-Criteria Decision Analysis*, vol. 7, no. 1, p. 34–47.

Dai, C., Y. Wang, et M. Ye. 2015. « A new multi-objective particle swarm optimization algorithm based on decomposition », *Information Sciences*, vol. 325, p. 541 – 557.

Das, I. et J. E. Dennis. 1998. « Normal-boundary intersection : A new method for generating the pareto surface in nonlinear multicriteria optimization problems », *SIAM Journal on Optimization*, vol. 8, no. 3, p. 631–657.

Davis, L. 1985. « Applying adaptive algorithms to epistatic domains ». In *Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 1*. Coll. « IJCAI'85 », p. 162–164, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

De Toro Negro, F., J. Ortega, E. Ros, S. Mota, B. Paechter, et J. M. Martin. 2004. « Psfga : Parallel processing and evolutionary computation for multiobjective optimisation », *Parallel Computing*, vol. 30, no. 5–6, p. 721–739.

Deb, K. 2001. *Multi-Objective Optimization Using Evolutionary Algorithms*. Coll. « Wiley Interscience Series in Systems and Optimization ». Wiley.

Deb, K. et T. Goel. 2001. *Controlled Elitist Non-dominated Sorting Genetic Algorithms for Better Convergence*. Coll. Zitzler, E., L. Thiele, K. Deb, C. Coello Coello, et D. Corne, éditeurs, Coll. « *Evolutionary Multi-Criterion Optimization* ». T. 1993, série *Lecture Notes in Computer Science*, p. 67–81. Springer Berlin Heidelberg.

Deb, K. et H. Jain. 2014. « 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*, vol. 18, no. 4, p. 577–601.

Deb, K., A. Pratap, S. Agarwal, et T. Meyarivan. 2002. « A fast and elitist multiobjective genetic algorithm : Nsga-ii », *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, p. 182–197.

Deb, K., P. Zope, et A. Jain. 2003. *Distributed Computing of Pareto-Optimal Solutions with Evolutionary Algorithms*. T. 2632, série *Lecture Notes in Computer Science*, book section 38, p. 534–549. Springer Berlin Heidelberg.

Delgado, M., M. P. Cuéllar, et M. del Carmen Pegalajar Jiménez. 2008. « Multiobjective hybrid optimization and training of recurrent neural networks », *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 38, no. 2, p. 381–403.

Delévacq, A., P. Delisle, M. Gravel, et M. Krajecki. 2013. « Parallel ant colony optimization on graphics processing units », *Journal of Parallel and Distributed Computing*, vol. 73, no. 1, p. 52 – 61. Metaheuristics on GPUs.

Depolli, M., R. Trobec, et B. Filipič. 2013. « Asynchronous master-slave parallelization of differential evolution for multi-objective optimization », *Evolutionary Computation*, vol. 21, no. 2, p. 261–291.

Doerner, K., W. Gutjahr, R. Hartl, C. Strauss, et C. Stummer. 2004. « Pareto ant colony optimization : A metaheuristic approach to multiobjective portfolio selection », *Annals of Operations Research*, vol. 131, no. 1-4, p. 79–99.

Dongarra, J. 1979. « Linpack : Users' guide ». In *Number 8. Society for IndustrialMathematics*.

Dorigo, M. et L. M. Gambardella. 1997. « Ant colony system : a cooperative learning approach to the traveling salesman problem », *IEEE Trans. Evolutionary Computation*, p. 53–66.

Dorronsoro, B., G. Danoy, A. J. Nebro, et P. Bouvry. 2013. « Achieving super-linear performance in parallel multi-objective evolutionary algorithms by means of cooperative coevolution », *Computers & OR*, p. 1552–1563.

Drugan, M. M. et D. Thierens. 2012. « Stochastic pareto local search : Pareto neighbourhood exploration and perturbation strategies », *Journal of Heuristics*, vol. 18, no. 5, p. 727–766.

Dubois-Lacoste, J., M. López-Ibáñez, et T. Stützle. 2015. « Anytime pareto local search », *European Journal of Operational Research*, vol. 243, no. 2, p. 369 – 385.

Durillo, J. J., A. J. Nebro, F. , et E. Alba. 2008. « A study of master-slave approaches to parallelize nsga-ii ». In *Parallel and Distributed Processing. IPDPS 2008. IEEE International Symposium on*, p. 1–8.

Durillo, J. J. et A. J. Nebro. 2011. « jmetal : A java framework for multi-objective optimization », *Advances in Engineering Software*, vol. 42, no. 10, p. 760 – 771.

Eberhart, R. et J. Kennedy. 1995. « Particle swarm optimization ». In *IEEE International Conference on Neural Networks*.

Ehrgott, M. et I. Winz. 2008. « Interactive decision support in radiation therapy treatmentplanning ». In *OR Spectrum 30*.

Essabri, A., M. Gzara, et T. Loukil. 2006. « Parallel multi-objective evolutionary algorithm with multi-front equitable distribution ». In *Grid and Cooperative Computing. GCC 2006. Fifth International Conference*, p. 241–244.

Fernández, A., C. Gil, R. Baños, et M. G. Montoya. 2013. « A parallel multi-objective algorithm for two-dimensional bin packing with rotations and load balancing », *Expert Systems with Applications*, vol. 40, no. 13, p. 5169 – 5180.

Ferringer, M., D. Spencer, et P. Reed. 2009. « Many-objective reconfiguration of operational satellite constellations with the large-cluster epsilon non-dominated sorting genetic algorithm-ii ». In *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, p. 340–349.

Figueira, José, R., A. Liefooghe, E.-G. Talbi, et P. Wierzbicki, Andrzej. 2010. « A parallel multiple reference point approach for multi-objective optimization », *European Journal of Operational Research*, vol. 205, no. 2, p. 390 – 400.

Flynn, M. J. 1972. « Some computer organizations and their effectiveness ». In *IEEE Trans. Comput.*, p. 948.

Fonseca, C. M. et P. J. Fleming. 1993. Genetic algorithms for multiobjective optimization : Formulation, discussion and generalization.

Foster, I. 1995. « Designing and building parallel programs : Concepts and tools for parallel software ». In *Addison-Wesley*.

Gagné C., Zinflou A., G. M. 2013. « Chapitre 2 : Stratégies de résolution pareto pour le problème industriel d'ordonnancement de voitures », *Métaheuristiques pour l'ordonnancement multicritère et les problèmes de transport : , Traité RTA, série Productique, Hermès Sciences, Lavoisier, Edited by B. Jarboui, P. Siarry and J. Teghem,*, p. 933–939.

Gagné, C. et M. Parizeau. 2006. « Genericity in evolutionary computation software tools : Principles and case-study », *International Journal on Artificial Intelligence Tools*, vol. 15, p. 173–194.

García-Martínez, C., O. Cordón, et F. Herrera. 2007. « A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria {TSP} », *European Journal of Operational Research*, vol. 180, no. 1, p. 116 – 148.

Garrett, D. et D. Dasgupta. 2009. « An empirical comparison of memetic algorithm strategies on the multiobjective quadratic assignment problem ». In *2009 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making(MCDM)*, p. 80–87.

Garza-Fabre, M., G. Pulido, et C. Coello. 2009. *Ranking Methods for Many-Objective Optimization*. Coll. Aguirre, A., R. Borja, et C. Garciá, éditeurs, Coll. « *MICAI 2009 : Advances in Artificial Intelligence* ». T. 5845, série *Lecture Notes in Computer Science*, p. 633–645. Springer Berlin Heidelberg.

Giagkiozis, I., R. Purshouse, et P. Fleming. 2014. « Generalized decomposition and cross entropy methods for many-objective optimization », *Information Sciences*, vol. 282, p. 363 – 387.

Goldberg, D. 1989. « Genetic algorithms in search, optimization and machine learning ». In *Addison-Wesley Longman Publishing Co*.

Golub, M. et D. Jakobovic. 2000. « A new model of global parallel genetic algorithm ». In *Information Technology Interfaces, 2000. ITI 2000. Proceedings of the 22nd International Conference on*, p. 363–368.

Gourisaria, M. K., B. S. P. Mishra, et S. Dehuri. 2013. « A hybrid parallel multi-objective genetic algorithm : Hybjaciscone model », *International Journal of Computer Applications*, vol. 66, no. 7, p. 1–6. Full text available.

Gupta, S. et G. Tan. 2015. « A scalable parallel implementation of evolutionary algorithms for multi-objective optimization on gpus ». In *2015 IEEE Congress on Evolutionary Computation (CEC)*, p. 1567–1574.

Hadoop. 2015. « Apache Hadoop 2.7.0 : documentation ». In *http ://hadoop.apache.org/docs/r2.7.0/*.

Haimes, Y., U. Lasdon, et D. Wismer. 1971. « On a bicriterion formulation of the problems of integrated system identification and system optimization », *Systems, Man and Cybernetics, IEEE Transactions on*, vol. SMC-1, no. 3, p. 296–297.

Handl, J. et J. Knowles. 2008. *Modes of Problem Solving with Multiple Objectives : Implications for Interpreting the Pareto Set and for Decision Making*. Coll. Knowles, J., D. Corne, K. Deb, et D. Chair, éditeurs, Coll. « *Multiobjective Problem Solving from Nature* ». Coll. « Natural Computing Series », p. 131–151. Springer Berlin Heidelberg.

Hansen, M. P. et A. Jaszkiewicz. 1998. Evaluating the quality of approximations to the non-dominated set.

Hernández Gómez, R., C. A. Coello Coello, et E. Alba. 2016. « A parallel version of sms-emoa for many-objective optimization problems ». In Handl, J., E. Hart, P. R. Lewis, M. López-Ibáñez, G. Ochoa, et B. Paechter, éditeurs, *Parallel Problem Solving from Nature – PPSN XIV*, p. 568–577, Cham. Springer International Publishing.

Hiroyasu, T., M. Miki, et S. Watanabe. 2000. « The new model of parallel genetic algorithm in multi-objective optimization problems - divided range multi-objective genetic algorithm ». In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*. T. 1, p. 333–340 vol.1.

Horn, J., J. Horn, N. Nafpliotis, N. Nafpliotis, D. E. Goldberg, et D. E. Goldberg. 1994. « A niched pareto genetic algorithm for multiobjective optimization ». In *In Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, p. 82–87.

Hughes, E. 2005. « Evolutionary many-objective optimisation : many once or one many ? ». In *Evolutionary Computation, 2005. The 2005 IEEE Congupress on*. T. 1, p. 222–227 Vol.1.

Ignizio, J. P. 1978. « A review of goal programming : A tool for multiobjective analysis », *The Journal of the Operational Research Society*, vol. 29, no. 11, p. pp. 1109–1119.

Iredi, S., D. Merkle, et M. Middendorf. 2001. *Bi-Criterion Optimization with Multi Colony Ant Algorithms*. Coll. Zitzler, E., L. Thiele, K. Deb, C. Coello Coello, et D. Corne, éditeurs, Coll. « *Evolutionary Multi-Criterion Optimization* ». T. 1993, série *Lecture Notes in Computer Science*, p. 359–372. Springer Berlin Heidelberg.

Ishibuchi, H. et T. Murata. 1998. « 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)*, vol. 28, no. 3, p. 392–403.

Ishibuchi, H., Y. Sakane, N. Tsukamoto, et Y. Nojima. 2011. « Implementation of cellular genetic algorithms with two neighborhood structures for single-objective and multi-objective optimization », *Soft Computing*, vol. 15, no. 9, p. 1749–1767.

Jaeggi, D., G. T. Parks, T. Kipouros, et P. J. Clarkson. 2008. « The development of a multi-objective Tabu Search algorithm for continuous optimisation problems », *European Journal of Operational Research*, vol. 185, p. 1192–1212.

Jagtap, S., S. Pani, et G. Shinde. 2011. « A hybrid parallel multi-objective genetic algorithm for 0/1 knapsack problem », *Journal of Software Engineering and Applications*, vol. 66, no. 4, p. 316–319.

Jaimes, A. L. et C. Coello. 2005. « Mrmoga : parallel evolutionary multiobjective optimization using multiple resolutions ». In *Evolutionary Computation. The 2005 IEEE Congress on*. T. 3, p. 2294–2301 Vol. 3.

Jaszkiewicz, A. 2002. « Genetic local search for multi-objective combinatorial optimization », *European Journal of Operational Research*, vol. 137, no. 1, p. 50 – 71.

JAVA. 2013. « Documentation en ligne ». In *http ://docs.oracle.com/javase/7/docs/api/*.

Karasakal, E. et A. Silav. 2015. « A multi-objective genetic algorithm for a bi-objective facility location problem with partial coverage », *TOP*, p. 1–27.

Ke, L., Q. Zhang, et R. Battiti. 2013. « Moea/d-aco : A multiobjective evolutionary algorithm using decomposition and antcolony », *IEEE Transactions on Cybernetics*, vol. 43, no. 6, p. 1845–1859.

Ke, L., Q. Zhang, et R. Battiti. 2014. « Hybridization of decomposition and local search for multiobjective optimization », *IEEE Transactions on Cybernetics*, vol. 44, no. 10, p. 1808–1820.

Khare, V., X. Yao, et K. Deb. 2003. *Performance Scaling of Multi-objective Evolutionary Algorithms*. Coll. Fonseca, C., P. Fleming, E. Zitzler, L. Thiele, et K. Deb, éditeurs, Coll. « *Evolutionary Multi-Criterion Optimization* ». T. 2632, série *Lecture Notes in Computer Science*, p. 376–390. Springer Berlin Heidelberg.

Knowles, J. et D. Corne. 2003. « Instance generators and test suites for the multiobjective quadratic assignment problem ». In Fonseca, C. M., P. J. Fleming, E. Zitzler, L. Thiele, et K. Deb, éditeurs, *Evolutionary Multi-Criterion Optimization*, p. 295–310, Berlin, Heidelberg. Springer Berlin Heidelberg.

Koopmans, T. C. et M. Beckmann. 1957. « Assignment problems and the location of economic activities », *Econometrica*, vol. 25, no. 1, p. 53–76.

Kronos. 2014. « Opencl, documentation en ligne ». In *www.khronos.org/opencl/*.

Lančinskas, A. et J. Žilinskas. 2012. *Approaches to Parallelize Pareto Ranking in NSGA-II Algorithm*. T. 7204, série *Lecture Notes in Computer Science*, book section 38, p. 371–380. Springer Berlin Heidelberg.

Larrañaga, P., C. Kuijpers, R. Murga, I. Inza, et S. Dizdarevic. 1999. « Genetic algorithms for the travelling salesman problem : A review of representations and operators », *Artificial Intelligence Review*, vol. 13, no. 2, p. 129–170.

Leite, N., R. Neves, N. Horta, F. Melício, et A. Rosa. 2015. *Solving a Capacitated Exam Timetabling Problem Instance Using a Bi-objective NSGA-II*. Coll. Madani, K., A. D. Correia, A. Rosa, et J. Filipe, éditeurs, Coll. « *Computational Intelligence* ». T. 577, série *Studies in Computational Intelligence*, p. 115–129. Springer International Publishing.

León, C., G. Miranda, E. Segredo, et C. Segura. 2009. *Parallel Hypervolume-Guided Hyperheuristic for Adapting the Multi-objective Evolutionary Island Model*. Coll. Krasnogor, N., M. Melián-Batista, J. Pérez, J. Moreno-Vega, et D. Pelta, éditeurs, Coll. « *Nature Inspired Cooperative Strategies for Optimization (NICSO 2008)* ». T. 236, série *Studies in Computational Intelligence*, p. 261–272. Springer Berlin Heidelberg.

Li, H. et D. Landa-Silva. 2009. « An elitist grasp metaheuristic for the multi-objective quadratic assignment problem ». In Ehrgott, M., C. M. Fonseca, X. Gandibleux, J.-K. Hao, et M. Sevaux, éditeurs, *Evolutionary Multi-Criterion Optimization*, p. 481–494, Berlin, Heidelberg. Springer Berlin Heidelberg.

Li, K., S. Kwong, Q. Zhang, et K. Deb. 2015. « Interrelationship-based selection for decomposition multiobjective optimization », *IEEE Transactions on Cybernetics*, vol. 45, no. 10, p. 2076–2088.

Li, K., Q. Zhang, S. Kwong, M. Li, et R. Wang. 2014. « Stable matching-based selection in evolutionary multiobjective optimization », *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 6, p. 909–923.

Li, X. 2003. « A non-dominated sorting particle swarm optimizer for multi-objective optimization », *Proceedings of the Genetic and Evolutionary Computation Conference. Lecture Notes in Computer Science*.

Li, Z., Y. Bian, R. Zhao, et J. Cheng. 2013. *A Fine-Grained Parallel Multi-objective Test Case Prioritization on GPU*. Coll. Ruhe, G. et Y. Zhang, éditeurs, Coll. « *Search Based Software Engineering* ». T. 8084, série *Lecture Notes in Computer Science*, p. 111–125. Springer Berlin Heidelberg.

Li, Z., Z. Zhu, S. Liu, et Z. Wang. 2010. *A Distance Sorting Based Multi-Objective Particle Swarm Optimizer and Its Applications*. Coll. Li, K., X. Li, S. Ma, et G. Irwin, éditeurs, Coll. « *Life System Modeling and Intelligent Computing* ». T. 98, série *Communications in Computer and Information Science*, p. 30–36. Springer Berlin Heidelberg.

Lis, J. et A. E. Eiben. 1996. « A Multi-Sexual Genetic Algorithm for Multiobjective Optimization ». In Fukuda, T. et T. Furuhashi, éditeurs, *Proceedings of the 1996 International Conference on Evolutionary Computation*, p. 59–64, Nagoya, Japan. IEEE.

Liu, J., H. Zhang, K. He, et S. Jiang. 2018. « Multi-objective particle swarm optimization algorithm based on objective space division for the unequal-area facility layout problem », *Expert Systems with Applications*, vol. 102, p. 179 – 192.

López-Ibáñez, M., L. Paquete, et T. Stützle. 2006. « Hybrid population-based algorithms for the bi-objective quadratic assignment problem », *Journal of Mathematical Modelling and Algorithms*, vol. 5, no. 1, p. 111–137.

Luna, F. et E. Alba. 2015. *Parallel Multiobjective Evolutionary Algorithms*. Coll. Kacprzyk, J. et W. Pedrycz, éditeurs, Coll. « *Springer Handbook of Computational Intelligence* », p. 1017–1031. Springer Berlin Heidelberg.

Luna, F., A. Nebro, et E. Alba. 2006. *Parallel Evolutionary Multiobjective Optimization*. T. 22, série *Studies in Computational Intelligence*, book section 2, p. 33–56. Springer Berlin Heidelberg.

Lust, T. et A. Jaszkiewicz. 2010. « Speed-up techniques for solving large-scale biobjective tsp », *Computers & Operations Research*, vol. 37, no. 3, p. 521 – 533. Hybrid Metaheuristics.

Lust, T. et J. Teghem. 2010a. « The multiobjective multidimensional knapsack problem : a survey and a new approach », *CoRR*, vol. abs/1007.4063.

Lust, T. et J. Teghem. 2010b. *The Multiobjective Traveling Salesman Problem : A Survey and a New Approach*, p. 119–141. Berlin, Heidelberg : Springer Berlin Heidelberg.

Lust, T. et J. Teghem. 2010c. « Two-phase pareto local search for the biobjective traveling salesman problem », *Journal of Heuristics*, vol. 16, no. 3, p. 475–510.

Mazière, F., P. Delisle, C. Gagné, et M. Krajecki. 2016a. Comparaison empirique de modèles parallèles pour l'algorithme multi-objectifs gismoo. Congrès annuel de la société Française de Recherche Opérationnelle et d'Aide à la Décision, ROADEF 2016, Compiègne.

Mazière, F., P. Delisle, C. Gagné, et M. Krajecki. 2016b. Étude comparative de modèles distribués pour la parallélisation de l'algorithme multi-objectifs gismoo. Conférence d'informatique en Parallélisme, Architecture et Système, Compas'16, Lorient.

Mazière, F., P. Delisle, C. Gagné, et M. Krajecki. 2018. APM-MOEA : An asynchronous parallel model for multi-objective evolutionary algorithms. International Conference on Metaheuristics and Nature Inspired Computing,Marrakech, Morocco.

McGeoch, L. A. 2001. 8th dimacs implementation challenge site, http ://dimacs.rutgers.edu/archive/challenges/tsp/.

Menchaca-Mendez, A. et C. A. C. Coello. 2015. « Gde-moea : A new moea based on the generational distance indicator and epsilon-dominance ». In *2015 IEEE Congress on Evolutionary Computation (CEC)*, p. 947–955.

Metaxioti, K. et K. Liagkouras. 2012. « Multiobjective evolutionary algorithms for portfolio management : A comprehensive literature review ». In *Expert Systems with Applications*.

Mezura-Montes, E. et C. Coello. 2008a. *Constrained Optimization via Multiobjective Evolutionary Algorithms*, book section 3, p. 53–75. Coll. « Natural Computing Series ». Springer Berlin Heidelberg.

Mezura-Montes, E. et C. C. Coello. 2008b. *Constrained Optimization via Multiobjective Evolutionary Algorithms*. Coll. « Natural Computing Series ». Springer Berlin Heidelberg.

Miettinen, K. 1999. *Nonlinear Multiobjective Optimization*. Coll. « International Series in Operations Research & Management Science ». Springer US.

Miner, D. et A. Shook. 2012. *MapReduce Design Patterns : Building Effective Algorithms and Analytics for Hadoop and Other Systems*. O'Reilly Media, Inc., 1st édition.

Mishra, B. S. P., S. Dehuri, R. Mall, et A. Ghosh. 2011. « Parallel single and multiple objectives genetic algorithms : A survey », *IJAEC*, vol. 2, p. 21–57.

Mora, A., P. García-Sánchez, J. Merelo, et P. Castillo. 2013. « Pareto-based multi-colony multi-objective ant colony optimization algorithms : an island model proposal », *Soft Computing*, vol. 17, no. 7, p. 1175–1207.

Moreno, J. J., G. Ortega, E. Filatovas, J. A. Martínez, et E. M. Garzón. 2017. « Using low-power platforms for evolutionary multi-objective optimization algorithms », *The Journal of Supercomputing*, vol. 73, no. 1, p. 302–315.

Morse, J. N. 1980. « Reducing the size of the nondominated set : Pruning by clustering », *Computers & Operations Research*, vol. 7, no. 1–2, p. 55–66.

Mostaghim, S., J. Branke, A. Lewis, et H. Schmeck. 2008. « Parallel multi-objective optimization using master-slave model on heterogeneous resources ». In *Evolutionary Computation. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, p. 1981–1987.

Mostaghim, S., J. Branke, et H. Schmeck. 2007. Multi-objective particle swarm optimization on computer grids.

MPI Forum. 2009. Message Passing Interface (MPI) Forum Home Page. `http://www.mpi-forum.org/`).

Munakata, T. et R. Barták. 2010. « Logic programming for combinatorial problems », *Artificial Intelligence Review*, vol. 33, no. 1, p. 135–150.

Nichols, B., D. Buttlar, et J. P. Farrell. 1996. *Pthreads Programming*. O'Reilly & Associates, Inc.

NVIDIA. 2007. « Cuda compute unified device architecture - programming guidee ». In *http ://docs.nvidia.com/cuda/cuda-c-programming-guide/*.

OpenMP. 2013. « Application Program Interface Version 4.0 ». In *http ://openmp.org/wp/openmp-specifications/*.

Pakhira, M. 2009. « A modified k-means algorithm to avoid empty clusters », vol. 1.

Paquete, L., M. Chiarandini, et T. Stützle. 2004. *Pareto Local Optimum Sets in the Biobjective Traveling Salesman Problem : An Experimental Study*. Coll. Gandibleux, X., M. Sevaux, K. Sörensen, et V. T'kindt, éditeurs, Coll. « *Metaheuristics for Multiobjective Optimisation* ». T. 535, série *Lecture Notes in Economics and Mathematical Systems*, p. 177–199. Springer Berlin Heidelberg.

Paquete, L. et T. Stützle. 2006. « A study of stochastic local search algorithms for the biobjective qap with correlated flow matrices », *European Journal of Operational Research*, vol. 169, no. 3, p. 943 – 959.

Paquete, L. et T. Stützle. 2009. « Design and analysis of stochastic local search for the multiobjective traveling salesman problem », *Computers & Operations Research*, vol. 36, no. 9, p. 2619 – 2631.

Paquete, L. et T. Stützle. 2003. « A two-phase local search for the biobjective traveling salesman problem ». In Fonseca, C. M., P. J. Fleming, E. Zitzler, L. Thiele, et K. Deb, éditeurs, *Evolutionary Multi-Criterion Optimization*, p. 479–493, Berlin, Heidelberg. Springer Berlin Heidelberg.

Pareto, V. 1876. « Cours d'economie politique », *F. Rouge*, vol. I et II.

Parsopoulos, K. E. et M. N. Vrahatis. 2002. « Particle swarm optimization method in multiobjective problems ». In *Proceedings of the 2002 ACM Symposium on Applied Computing*. Coll. « SAC '02 », p. 603–607, New York, NY, USA. ACM.

Pilat, M. et R. Neruda. 2010. « Combining multiobjective and single-objective genetic algorithms in heterogeneous island model ». In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, p. 1–8.

Potter, M. A. et K. A. D. Jong. 1994. A cooperative coevolutionary approach to function optimization.

Punnen, A. P. 2007. *The Traveling Salesman Problem : Applications, Formulations and Variations*, p. 1–28. Boston, MA : Springer US.

Qingfu, Z. et L. Hui. 2007. « Moea/d : A multiobjective evolutionary algorithm based on decomposition », *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 6, p. 712–731.

Qiu, X., Y. Huang, et K. Tan. 2015. *A Novel Multi-objective Optimization Framework Combining NSGA-II and MOEA/D*. Coll. Handa, H., H. Ishibuchi, Y.-S. Ong, et K.-C. Tan, éditeurs, Coll. « *Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems - Volume 2* ». T. 2, série *Proceedings in Adaptation, Learning and Optimization*, p. 227–237. Springer International Publishing.

Quinn, M. J. 2003. *Parallel Programming in C with MPI and OpenMP*. McGraw-Hill Education Group.

Rauber, T. et G. Rünger. 2010. *Parallel Programming - for Multicore and Cluster Systems.*

Reinelt, G. 1990. TSPLIB - A T.S.P. Library. Rapport no. 250, Universität Augsburg, Institut für Mathematik, Augsburg.

ROMEO. 2018. « Romeo hpc center ». In *https ://romeo.univ-reims.fr/.*

Samanta, S., D. Philip, et S. Chakraborty. 2018. « Bi-objective dependent location quadratic assignment problem : Formulation and solution using a modified artificial bee colony algorithm », *Computers & Industrial Engineering*, vol. 121, p. 8 – 26.

Sanhueza, C., F. Jimenez, R. Berretta, et P. Moscato. 2017. « Pasmoqap : A parallel asynchronous memetic algorithm for solving the multi-objective quadratic assignment problem ». In *2017 IEEE Congress on Evolutionary Computation, CEC 2017, Donostia, San Sebastián, Spain, June 5-8, 2017*, p. 1103–1110.

Sato, H. 2015. « Analysis of inverted pbi and comparison with other scalarizing functions in decomposition based moeas », *Journal of Heuristics*, vol. 21, no. 6, p. 819–849.

Schaffer, J. 1985. « Multiple objective optimization with vector evaluated genetic algorithms ». In *ICGA'85*, p. 93–100.

Seada, H. et K. Deb. 2015. *U-NSGA-III : A Unified Evolutionary Optimization Procedure for Single, Multiple, and Many Objectives : Proof-of-Principle Results*. Coll. Gaspar-Cunha, A., C. Henggeler Antunes, et C. C. Coello, éditeurs, Coll. « *Evolutionary Multi-Criterion Optimization* ». T. 9019, série *Lecture Notes in Computer Science*, p. 34–49. Springer International Publishing.

Sülflow, A., N. Drechsler, et R. Drechsler. 2007. *Robust Multi-Objective Optimization in High Dimensional Spaces*. Coll. Obayashi, S., K. Deb, C. Poloni, T. Hiroyasu, et T. Murata, éditeurs, Coll. « *Evolutionary Multi-Criterion Optimization* ». T. 4403, série *Lecture Notes in Computer Science*, p. 715–726. Springer Berlin Heidelberg.

Steinhaus, H. 1956. « Sur la division des corps matériels en parties », *Bulletin de l'Académie Polonaise des Sciences*, vol. Cl. III — Vol. IV, no. 12, p. 801–804.

Stewart, T., O. Bandte, H. Braun, N. Chakraborti, et M. Ehrgott. 2008. « Real-world applications of multiobjective optimization ». In *Multiobjective Optimization,*.

Streichert, F., H. Ulmer, et A. Zell. 2005. *Parallelization of Multi-objective Evolutionary Algorithms Using Clustering Algorithms*. T. 3410, série *Lecture Notes in Computer Science*, book section 7, p. 92–107. Springer Berlin Heidelberg.

Strohmaier, E. 2018. TOP500 Supercomputer Site.

Taillard, E. D. 1995. « Comparison of iterative searches for the quadratic assignment problem », *Location Science*, vol. 3, no. 2, p. 87 – 105.

Talbi, E.-G. 2000. Métaheuristiques pour l'optimisation combinatoire multi-objectif : Etat de l'art.

Talbi, E.-G. 2009. *Metaheuristics : From Design to Implementation*. Wiley Publishing.

Talbi, E.-G. 2018. « A unified view of parallel multi-objective evolutionary algorithms », *Journal of Parallel and Distributed Computing*, p. –.

Talbi, E.-G., S. Mostaghim, T. Okabe, H. Ishibuchi, G. Rudolph, et C. A. Coello Coello. 2008. *Parallel Approaches for Multiobjective Optimization*, p. 349–372. Berlin, Heidelberg : Springer Berlin Heidelberg.

Tanenbaum, A. S. 2005. *Structured Computer Organization (5th Edition)*. Upper Saddle River, NJ, USA : Prentice-Hall, Inc.

TBB. 2006. « Threading building blocks, documentation en ligne ». In *https ://www.threadingbuildingblocks.org/documentation*.

Tian, Y., X. Zhang, R. Cheng, et Y. Jin. 2016. « A multi-objective evolutionary algorithm based on an enhanced inverted generational distance metric ». In *2016 IEEE Congress on Evolutionary Computation (CEC)*, p. 5222–5229.

Tripathi, K., S. Bandyopadhyay, et S. K. Pal. 2007. « Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients ». In *Information Sciences*.

Trivedi, A., D. Srinivasan, K. Sanyal, et A. Ghosh. 2017. « A survey of multiobjective evolutionary algorithms based on decomposition », *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 3, p. 440–462.

Van Veldhuizen, D. A. et G. B. Lamont. 2000. « Multiobjective evolutionary algorithms : Analyzing the state-of-the-art », *Evolutionary Computation*, vol. 8, no. 2, p. 125–147.

Veldhuizen, D. A. V., J. B. Zydallis, et G. B. Lamont. 2003. « Considerations in engineering parallel multiobjective evolutionary algorithms », *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, p. 144–173.

Wang, L., Q. Zhang, A. Zhou, M. Gong, et L. Jiao. 2016. « Constrained subproblems in a decomposition-based multiobjective evolutionary algorithm », *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, p. 475–480.

White, T. et S. He. 2012. *An Empirical Comparison of Several Recent Multi-objective Evolutionary Algorithms*. Coll. Iliadis, L., I. Maglogiannis, et H. Papadopoulos, éditeurs, Coll. « *Artificial Intelligence Applications and Innovations* ». T. 381, série *IFIP Advances in Information and Communication Technology*, p. 48–57. Springer Berlin Heidelberg.

Yedla, M., S. R. Pathakota, et T. M. Srinivasa. 2010. « Enhancing k-means clustering algorithm with improved initial centre », *International Journal of Computer Science and Information Technologies*, p. 121–125.

Yu, W.-J., J.-Z. Li, W.-N. Chen, et J. Zhang. 2017. « A parallel double-level multiobjective evolutionary algorithm for robust optimization », *Applied Soft Computing*, vol. 59, p. 258 – 275.

Yuan, Y., H. Xu, et B. Wang. 2014. « An improved nsga-iii procedure for evolutionary many-objective optimization ». In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*. Coll. « GECCO '14 », p. 661–668, New York, NY, USA. ACM.

Zhang, W., M. Gen, et J. Jo. 2014. « Hybrid sampling strategy-based multiobjective evolutionary algorithm for process planning and scheduling problem », *Journal of Intelligent Manufacturing*, vol. 25, no. 5, p. 881–897.

Zhao, G., J. Perilla, E. Yufenyuy, X. Meng, B. Chen, J. Ning, J. Ahn, A. M Gronenborn, K. Schulten, C. Aiken, et P. Zhang. 2013. « Mature hiv–1 capsid structure by cryo–electron microscopy and all–atom molecular dynamics », vol. 497, p. 643–646.

Zhou, A., B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, et Q. Zhang. 2011. « Multiobjective evolutionary algorithms : A survey of the state of the art », *Swarm and Evolutionary Computation*, vol. 1, no. 1, p. 32 – 49.

Zhou, Y. et Y. Tan. 2009. « Gpu-based parallel particle swarm optimization ». In *Proceedings of the Eleventh Conference on Congress on Evolutionary Computation*. Coll. « CEC'09 », p. 1493–1500, Piscataway, NJ, USA. IEEE Press.

Zinflou, A., C. Gagné, et M. Gravel. 2013. « A hybrid genetic/immune strategy to tackle the multiobjective quadratic assignment problem », *The 2018 Conference on Artificial Life : A Hybrid of the European Conference on Artificial Life (ECAL) and the International Conference on the Synthesis and Simulation of Living Systems (ALIFE)*, p. 933–939.

Zinflou, A., C. Gagné, M. Gravel, et W. Price. 2008. « Pareto memetic algorithm for multiple objective optimization with an industrial application », *Journal of Heuristics*, vol. 14, no. 4, p. 313–333.

Zinflou, A., C. Gagné, et M. Gravel. 2012. « Gismoo : A new hybrid genetic/immune strategy for multiple-objective optimization », *Computers & Operations Research*, vol. 39, no. 9, p. 1951–1968.

Zitzler, E. 1999. « Evolutionary algorithms for multiobjective optimization : Methods and applications ». Thèse de Doctorat.

Zitzler, E., K. Deb, et L. Thiele. 2000. « Comparison of multiobjective evolutionary algorithms : Empirical results », *Evolutionary Computation*, vol. 8, no. 2, p. 173–195.

Zitzler, E. et S. Künzli. 2004. *Indicator-Based Selection in Multiobjective Search*. Coll. Yao, X., E. Burke, J. Lozano, J. Smith, J. Merelo-Guervós, J. Bullinaria, J. Rowe, P. Tiňo, A. Kabán, et H.-P. Schwefel, éditeurs, Coll. « *Parallel Problem Solving from Nature - PPSN VIII* ». T. 3242, série *Lecture Notes in Computer Science*, p. 832–842. Springer Berlin Heidelberg.

Zitzler, E., M. Laumanns, et L. Thiele. 2002a. « Spea2 : Improving the strength pareto evolutionary algorithm for multiobjective optimization ». In *Evolutionary Methods for Design, Optimisation, and Control*, p. 95–100. CIMNE, Barcelona, Spain.

Zitzler, E., M. Laumanns, L. Thiele, C. M. Fonseca, et V. G. da Fonseca. 2002b. « Why quality assessment of multiobjective optimizers is difficult ». In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*. Coll. « GECCO'02 », p. 666–674, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Zitzler, E. et L. Thiele. 1998. *Multiobjective optimization using evolutionary algorithms — A comparative case study*. T. 1498, série *Lecture Notes in Computer Science*, book section 29, p. 292–301. Springer Berlin Heidelberg.

Özkale, C. et A. Fığlalı. 2013. « Evaluation of the multiobjective ant colony algorithm performances on biobjective quadratic assignment problems », *Applied Mathematical Modelling*, vol. 37, no. 14, p. 7822 – 7838.

Modèles de parallélisme pour les métaheuristiques multi-objectifs

Comme ils requièrent généralement des capacités de calcul importantes pour explorer de grandes régions de l'espace de recherche dans la résolution de problèmes réels et complexes, les algorithmes évolutionnaires multi-objectifs peuvent bénéficier des nouvelles architectures de calcul haute performance. Ainsi, des approches parallèles ont été proposées à la fois pour réduire leur temps d'exécution et améliorer la qualité des solutions fournies. Bien que des progrès significatifs aient été réalisés ces dernières années dans la conception et l'amélioration de modèles parallèles pour les algorithmes évolutionnaires, la plupart de ces modèles ont une extensibilité limitée et des difficultés à manipuler des problèmes variés. Résoudre efficacement un problème combinatoire multi-objectifs sur une architecture de calcul haute performance demeure encore un défi aujourd'hui.

Dans cette thèse, des mécanismes sont proposés au sein d'un nouveau modèle parallèle pour exploiter de multiples unités de calcul dans la résolution de problèmes multi-objectifs. L'approche générale reprend les principes des modèles en îles avec une division de l'espace objectif qui repose sur une méthode de regroupement. Les principales caractéristiques du modèle sont (i) la vue globale donnée à l'île organisatrice pour atteindre une meilleure distribution des solutions (ii) des communications asynchrones pour réduire les surcoûts du parallélisme (iii) des îles témoins pour accroitre la diversification (iv) une phase de recherche locale pour améliorer la qualité des solutions. Une étude expérimentale minutieuse est effectuée afin d'évaluer les performances de l'approche et plus particulièrement de chaque composante dans la résolution de deux problèmes combinatoires classiques, le problème de voyageur de commerce multi-objectifs et le problème d'affectation quadratique multi-objectifs. L'extensibilité et la qualité des solutions sont analysées comparativement aux modèles parallèles de la littérature.

Mots clés : Métaheuristiques, Multi-objectifs, Modèles parallèles, Algorithmes évolutionnaires, Optimisation combinatoire

Parallelism models for multi-objective metaheuristics

As they often require a high amount of computing resources to explore large portions of the search space and handle complex real-life constraints, multi-objective evolutionary algorithms could greatly benefit from today's high-performance computing architectures. Although significant progress has been made in recent years in the design and improvement of parallel models for evolutionary algorithms, most of these models have limited scalability and ability to solve various problems. In fact, solving multi-objective combinatorial optimization problems efficiently on a large number of processors remains a challenge today.

This thesis aims to propose an island model which is based on objective space division. The main features of the proposed model are the following (i) An organizer has a global view of the current search via a global archive (ii) Asynchronous cooperation between islands, especially for the exchange of local archives with the organizer to limit model overheads (iii) Control islands to guide the exploration of the search space and improve diversity (iv) A periodic use of a specific local search procedure to improve convergence. Extensive experiments have been conducted to evaluate the performance of the approach and more particularly of each component in the resolution of two classical combinatorial problems, the travelling salesman problem and quadratic assignment problem. Extensibility and quality of the solutions are analyzed compared to state-of-the-art parallel models.

Keywords :.Metaheuristics, Multi-Objective, Parallel models, Evolutionary algorithms, Combinatorial optimization

Discipline : INFORMATIQUE

UQAC

Université du Québec à Chicoutimi -

Département d'informatique et de mathématique

555 boul. de l'Université, Chicoutimi, Québec, Canada, G7H 2B1

écoles .urca Doctorales

Université de Reims Champagne-Ardenne –

CRESTIC - EA 3804

UFR Sciences Exactes et Naturelles, Moulin de la Housse, 51867 Reims