Contributions à l'optimisation multi-objectif à base de décomposition

Geoffrey Pruvost

Université de Lille
Inria Lille - Nord Europe

École doctorale **MADIS-631, Mathématiques-Sciences du numérique et de leurs interactions**
Unité de recherche **CRIStAL**

Thèse présentée par **Geoffrey Pruvost**

Soutenue le **3 décembre 2021**

En vue de l'obtention du grade de docteur de l'Université de Lille

Discipline **Informatique**
Spécialité **Optimisation**

Contributions à l'optimisation multi-objectif à base de décomposition

Thèse dirigée par Bilel Derbel directeur
Arnaud Liefooghe co-encadrant

Composition du jury

Rapporteurs	Idoumghar Lhassane	professeur à l'Université de Haute-Alsace	
	Evelyne Lutton	directeur de recherche à l'INRAE-AgroParisTech	
Examinateurs	Laurence Duchien	professeur à l'Université de Lille	président du jury
	Adrien Goeffon	MCF à l'Université d'Angers	
Directeurs de thèse	Bilel Derbel	MCF à l'Université de Lille	
	Arnaud Liefooghe	MCF à l'Université de Lille	

Contributions à l'optimisation multi-objectif à base de décomposition

Résumé

Dans cette thèse, nous nous intéressons à l'optimisation combinatoire multi-objectif, et en particulier aux algorithmes évolutionnaires à base de décomposition. Ce type d'approches consiste à décomposer le problème multi-objectif original en plusieurs sous-problèmes mono-objectifs, qui sont alors résolus de façon coopérative. Dans ce cadre, nous considérons la conception et l'analyse de nouveaux composants algorithmiques contribuant à la mise en place des fondations d'un framework d'optimisation à base de décomposition pour les problèmes combinatoires multi-objectifs dits "boîte-noire", pour lesquels la forme analytique des fonctions objectif n'est pas connue de l'algorithme de résolution. Tout d'abord, nous étudions les éléments clés pour une meilleure répartition des efforts de calculs tout au long du processus d'optimisation. Pour cela, nous étudions l'impact conjoint de la taille de la population et du nombre de solutions générées par itération, tout en proposant différentes stratégies de sélection du ou des sous-problèmes à optimiser à chaque étape. Nous étudions ensuite différents mécanismes permettant de s'échapper des optima locaux. Ceux-ci s'inspirent de techniques issues de l'optimisation mono-objectif et permettent d'améliorer considérablement le profil de convergence des approches considérées. Nous nous plaçons pour finir dans un contexte d'optimisation coûteuse, où l'évaluation de chaque solution s'avère particulièrement gourmande en temps de calcul, ce qui limite considérablement le budget alloué à l'optimisation. Pour cela, nous étudions de nouveaux composants s'appuyant sur des méta-modèles combinatoires, et nous considérons leur intégration au sein d'approches évolutionnaires multi-objectifs basées sur la décomposition.

Mots clés : optimisation multi-objectif, optimisation combinatoire, décomposition, moea/d, méta-modèle, optimum local, algorithme évolutionnaire

---

Abstract

In this thesis, we are interested in multi-objective combinatorial optimization, and in particular in evolutionary algorithms based on decomposition. This type of approaches consists in decomposing the original multi-objective problem into multiple single-objective sub-problems that are then solved cooperatively. In this context, we consider the design and the analysis of new algorithmic components contributing to the establishment of the foundations of an optimization framework based on decomposition for multi-objective combinatorial problems known as "black box", i.e., for which the analytical form of the objective functions is not available to the solving algorithm. First of all, we investigate the key components for a better distribution of the computational efforts during the optimization process. To this end, we study the joint impact of the population size and of the number of solutions generated at each iteration, while proposing different strategies for selecting one ore multiple sub-problem(s) to be optimized at each stage. We then study different mechanisms allowing to escape from local optima. They are inspired by techniques from single-objective optimization, and we show they can significantly improve the convergence profile of the considered approaches. Finally, we consider the context of expensive optimization, where the evaluation of each solution is particularly intensive in terms of computational resources. This hence drastically restrict the budget allocated to the optimization process. As such, we investigate new components based on combinatorial meta-models, and we consider their integration within decomposition-based multi-objective evolutionary approaches.

Keywords: multi-objective optimisation, combinatorial optimization, decomposition, moea/d, surrogate, local optima, evolutionary algorithm

---

CRIStAL
Université de Lille - Campus scientifique – Bâtiment ESPRIT – Avenue Henri Poincaré – 59655 Villeneuve d'Ascq – France

Table des matières

Introduction

Cette thèse a été conduite dans le cadre du projet *BigMO* (*Big Multi-objective Optimization*), financé par l'Agence Nationale de la Recherche (ANR), en coopération avec *City University of Hong Kong*. Elle a été réalisée au sein de l'équipe-projet Bonus (*Big Optimization aNd Ultra Scale computing*), commune au Centre de Recherche en Informatique, Signal et Automatique de Lille (CRIStAL, UMR 9189) de l'Université de Lille, et le centre de recherche Inria Lille-Nord Europe.

Contexte et motivations. Les décisions que nous prenons le sont en fonction des connaissances que l'on a pu acquérir, de notre expérience, du contexte dans lequel on se trouve, et de notre analyse du problème rencontré. Dès le XVIIe siècle, des outils d'aide à la décision ont été mis en place en se basant sur différents concepts et théories. Certaines techniques vont prendre le nom de recherche opérationnelle que bien plus tard, durant la seconde guerre mondiale, avec par exemple la volonté d'optimiser des problèmes d'implantation optimale de radars de surveillance, ou encore la gestion de convois d'approvisionnement. Optimiser un problème consiste à rechercher la meilleure solution réalisable en fonction d'un ou plusieurs critères d'évaluation (les objectifs). Pour entreprendre la résolution d'un problème d'optimisation, on doit préalablement définir ce qu'est une solution au problème que l'on souhaite résoudre, mais également avoir la capacité d'évaluer cette solution, c'est-à-dire de lui donner une valeur pour quantifier sa qualité ou son coût. Une solution à un problème d'optimisation est constituée d'un ensemble de variables dont les valeurs permettent d'évaluer sa qualité. Il va de soi que la difficulté à identifier de bonnes solutions dépend du nombre de variables, ainsi que du nombre d'objectifs à prendre en compte.

L'évolution spectaculaire des ressources de calculs nous permet chaque jour de franchir de nouvelles limites. En particulier, cette évolution pousse de plus en plus loin les capacités des algorithmes d'optimisation, en leur permettant notamment de disposer d'une puissance de calcul remarquable, permettant par exemple d'explorer

et d'évaluer un espace de solutions toujours plus grand. Malgré cela, la complexité de certains problèmes présente toujours des difficultés intrinsèques aux ingénieurs et aux chercheurs, notamment quand le nombre d'objectifs ou le nombre de variables est important. Dans certains cas, il est possible d'utiliser les propriétés connues du problème à résoudre afin d'orienter plus facilement la recherche des solutions optimales, c'est ce que l'on appelle l'optimisation boite-blanche ou boite-grise. Dans d'autres cas, les propriétés permettant de faciliter l'optimisation du problème ne sont pas connues, les fonctions d'évaluation peuvent typiquement faire appel à des ressources externes qui ne permettent pas de connaître la forme analytique des fonctions. Ces problèmes sont alors appelés problèmes "boite-noire" ou "black-box".

De nombreuses techniques existent pour résoudre ce type de problèmes, telles que les algorithmes évolutionnaires qui se basent sur le principe de la théorie de l'évolution. Il s'agit de faire évoluer un ensemble de solutions constituant ce que l'on appelle une population par analogie aux processus d'évolution que l'on peut rencontrer dans la nature. La première phase des algorithmes évolutionnaires est de sélectionner, dans la population, des solutions parentes pouvant servir à la génération de nouvelles solutions. Ces solutions parentes vont ensuite être soumises à des opérateurs de variation, aussi appelés opérateurs génétiques. Faisant appel à la métaphore de l'évolution d'individus dans la nature, les opérateurs génétiques désignent des composants ayant la capacité de générer de nouvelles solutions candidates en utilisant des techniques algorithmiques inspirées de la biologie comme les croisements et les mutations. Une fois ces solutions candidates générées, un processus de remplacement de la population entre en jeu. Il permet en général de remplacer les solutions de la population par les meilleures solutions candidates ainsi générées. Ce processus représente ce que l'on appelle une génération. Il est ensuite répété un certain nombre de fois jusqu'à ce qu'une condition d'arrêt soit satisfaite, par exemple un nombre maximal de générations ou un critère de qualité de la population.

Les algorithmes évolutionnaires conviennent parfaitement aux problèmes dits multi-objectifs. Dans ce cas, nous cherchons à optimiser plusieurs fonctions objectifs, souvent contradictoires. Ceci rend donc impossible l'identification d'une solution optimale unique satisfaisant tous les objectifs à la fois. Une difficulté principale des problèmes multi-objectifs est la manière de déterminer si une solution est supérieure à une autre solution. Pour cela, la relation de dominance Pareto est utilisée. On considère en effet qu'une solution x est dominée par une solution y si tous les objectifs de y sont au moins égaux à ceux de x, et qu'au moins un objectif de y est strictement meilleur que celui de x. Dans ce cadre, les algorithmes évolutionnaires permettent de façon naturelle

de faire évoluer une population vers l'ensemble des solutions non-dominées, aussi appelé l'ensemble Pareto. Cet ensemble permet aux décideurs de disposer des meilleurs compromis possibles par rapport aux différents objectifs, constituant ce que l'on appelle le front Pareto.

Objectifs et contributions. Il existe plusieurs approches pour résoudre un problème multi-objectif avec un algorithme évolutionnaire. Les approches existantes se divisent en général suivant trois classes : (1) les approches basées sur la dominance utilisent la dominance de Pareto pour comparer et maintenir la population de solutions, (2) les approches basées sur les indicateurs font évoluer leur population en fonction d'un indicateur renseignant sur la qualité d'un ensemble de solutions, et (3) les approches basées sur l'agrégation ou la décomposition des fonctions objectifs. Ces trois classes ont pu montrer leur efficacité pour résoudre un panorama relativement large de scénarios d'optimisation et de problèmes complexes, et font l'objet d'améliorations et d'études depuis plusieurs années. Dans nos travaux, nous nous intéressons spécifiquement aux approches basées sur la décomposition, à travers l'algorithme MOEA/D, un algorithme état-de-l'art dans le domaine.

L'algorithme MOEA/D est souvent considéré comme un framework général constitué de plusieurs composants et pouvant se décliner suivant plusieurs variantes. L'idée principale de MOEA/D est d'utiliser une fonction d'agrégation scalaire pour décomposer le problème original en plusieurs sous-problèmes mono-objectifs, qui sont ensuite résolus de façon coopérative en utilisant un processus évolutionnaire. Dans cette thèse, nous nous intéressons spécifiquement à enrichir la conception du framework MOEA/D, et à en étudier les propriétés pour des problèmes d'optimisation combinatoire, c'est-à-dire des problèmes avec des variables de décision discrètes. Il est à noter que, relativement aux nombreux travaux traitant des problèmes à variables continues, le framework MOEA/D a été peu étudié en optimisation combinatoire.

Dans ce contexte, l'objectif premier de cette thèse est de contribuer à la mise en place des fondations d'un framework de décomposition pour l'optimisation combinatoire multi-objectif de type "boite-noire". Pour ce faire, nous nous intéressons à la conception et à l'étude de différents composants clés du framework MOEA/D. Plus précisément, dans une première contribution, nous considérons la problématique de la définition et du choix des sous-problèmes mono-objectifs à optimiser à chaque génération du processus évolutionnaire. Ceci est souvent étudié en optimisation continue, où des mécanismes dédiés ont été proposés en s'appuyant sur l'identification des sous-problèmes les plus "difficiles". Nous révisons alors les mécanismes état-de-l'art existants dans un

contexte d'optimisation combinatoire, et nous nous intéressons au comportement "anytime" de MOEA/D, c'est-à-dire la capacité de l'algorithme à converger rapidement vers un ensemble de solutions de bonne qualité. Dans une deuxième contribution, nous nous intéressons à la conception de mécanismes d'échappement aux optima locaux au sein de MOEA/D. Il s'agit en effet d'un aspect relativement peu étudié dans la communauté. Nos travaux montrent que le framework MOEA/D permet d'intégrer de façon simple et flexible des mécanismes issus de l'optimisation mono-objectif grâce au principe de décomposition sous-jacent. L'objectif premier de cette contribution est de permettre la mise en place de mécanismes de détection des optima locaux, pour ensuite proposer des méthodes d'échappements efficaces au cœur même de l'algorithme MOEA/D. Enfin, notre troisième et dernière contribution se situe dans un contexte d'optimisation coûteuse, c'est-à-dire quand le processus d'évaluation de la qualité d'une solution est coûteux en temps et/ou en ressources de calcul, limitant ainsi de façon drastique le nombre d'appels aux fonctions d'évaluations du processus d'optimisation. Cette limitation implique l'intégration à MOEA/D de composants qui permettent d'atteindre les meilleures solutions avec le moins d'évaluations possibles. Dans ce contexte, et alors que la plupart des recherches dans ce domaine considèrent des problèmes continus, nous nous attaquons à l'intégration de méta-modèles de substitution pour l'optimisation combinatoire multi-objectif.

Plan du manuscrit. Cette thèse est présentée en cinq chapitres résumés brièvement ci-dessous :

— Le chapitre 1 propose un aperçu de l'état-de-l'art pour l'optimisation multi-objectif combinatoire par décomposition. On y présente les fondements des algorithmes multi-objectifs par décomposition, leur intérêt et le comportement des différents composants utilisés dans ce type d'algorithmes. On présente également les différents problèmes d'optimisation utilisés comme benchmark tout au long de cette thèse en expliquant leurs spécificités.

— Le chapitre 2 propose une étude du composant de sélection des sous-problèmes à optimiser à chaque génération, dans un contexte combinatoire. On y montre en particulier que ce type de composant est important pour améliorer le comportement "anytime" de MOEA/D. Les trois paramètres affectant ce composant sont étudiés de manière à fournir une meilleure compréhension de leur configuration optimale en fonction de la difficulté et des caractéristiques des problèmes étudiés.

— Le chapitre 3 propose l'intégration d'un nouveau composant pour échapper aux optima locaux en optimisation multi-objectif par décomposition. L'intérêt

de cette étude est d'intégrer, au sein d'un algorithme de décomposition, des techniques d'échappement issues de l'optimisation mono-objectif. L'approche proposée dans ce cadre s'articule autour de deux aspects permettant respectivement d'étudier une technique de détection des optima locaux et différentes stratégies de perturbation de la population.

— Le chapitre 4 propose une utilisation de modèles de substitution basés sur les fonctions de Walsh discrètes. Nous y proposons une méthodologie pour l'optimisation coûteuse dans un contexte combinatoire et multi-objectif, puis nous présentons les différents composants algorithmiques clés. Nous nous attachons ensuite à analyser les différents composants proposés, afin d'étudier de façon précise leur impact conjoint. À notre connaissance, il s'agit ici des toutes premières études sur le sujet.

— Le dernier chapitre conclue la thèse avec un résumé des contributions essentielles, et un aperçu des questions et pistes de recherche soulevées par nos travaux.

Chapitre 1

Optimisation combinatoire multi-objectif par décomposition

Introduction

Le premier chapitre a pour but d'introduire les notions nécessaires à une bonne compréhension de cette thèse. Nous introduisons tout d'abord l'optimisation combinatoire multi-objectif en rappelant quelques définitions. Ensuite, nous présentons les principales méthodes de résolution pour l'optimisation multi-objectif, avec un intérêt particulier pour les approches basées sur la décomposition. Nous décrivons également la bibliothèque logicielle dédiée [81] que nous avons développée au cours de la thèse. Enfin, nous présentons les différentes méthodes d'analyse de performance et de comparaisons d'algorithmes que nous considérons tout au long du manuscrit, avec une présentation des problèmes d'optimisation étudiés dans nos campagnes expérimentales.

1.1 Optimisation combinatoire multi-objectif

1.1.1 Problèmes d'optimisation multi-objectif

L'optimisation combinatoire consiste à trouver la solution optimale parmi un ensemble discret de solutions appelé *espace de décision* ou *espace de recherche*, que l'on note X. L'espace de recherche est associé à un ou plusieurs critères d'évaluation, également appelé *objectif*, pour former ce que l'on appelle un problème d'optimisation. Lorsque plusieurs objectifs sont à optimiser conjointement, nous nous trouvons alors face à un *problème d'optimisation multi-objectif*. Nous pouvons définir l'espace objectif Z comme étant l'ensemble des vecteurs objectif réalisables où chaque solution $x \in X$

est associé à un vecteur objectif $z \in Z$. Les *fonctions objectifs*, permettent de donner une valeur qui définira la qualité des solutions présentes dans l'espace de recherche. Un problème d'optimisation multi-objectif est composé de plusieurs fonctions objectifs à optimiser de façon simultanée que l'on note $F = (f_1, f_2, \ldots, f_m)$, où m représente le nombre d'objectifs. Un problème d'optimisation multi-objectif peut être formulé de la façon suivante :

$$\begin{aligned} &\min F(x) = (f_1(x), \ldots, f_m(x)) \\ &\text{tel que } x \in X \end{aligned} \tag{1.1}$$

Une fonction $f(x) : X \to Z$ est appelée *fonction boite-noire* lorsqu'on ne connaît pas sa forme analytique. Si les fonctions f_i d'un problème multi-objectif sont de type *boite-noire*, alors on considère que le problème d'optimisation est un *problème boite-noire*. On s'intéresse dans ce manuscrit aux problèmes d'optimisation combinatoire multi-objectif *boite-noire*. Le fait qu'un problème d'optimisation soit de type *boite-noire* est impactant pour sa résolution, comme on le verra dans la prochaine section.

Contrairement à l'optimisation continue où chaque variable de l'espace de décision X est continue, les variables en optimisation combinatoire sont discrètes. Ces variables peuvent par exemple représenter un ordre d'éléments avec une permutation, comme pour le problème du voyageur de commerce [45]. Elles peuvent également se trouver sous une forme binaire (ou booléenne) pour définir si un élément est inclus ou non dans la solution, comme pour le problème du sac-à-dos [48]. Elles peuvent aussi représenter des valeurs quantitatives ou qualitatives grâce à des valeurs entières. Avec un espace de décision discret, il est donc possible en théorie de lister toutes les solutions possibles de l'espace de recherche si on considère avoir des ressources de calcul illimitées. En pratique, cette approche n'est pas possible car même si l'espace de décision est discret, il est de trop grande taille pour pouvoir énumérer toutes les solutions en un temps raisonnable. Nous nous intéressons dans cette thèse aux problèmes d'optimisation combinatoire considérés comme *NP-difficile* [77]. En optimisation combinatoire, beaucoup de problèmes sont NP-difficiles, et ne permettent pas d'être résolus de façon exacte en un temps de calcul raisonnable. Si un problème est NP-difficile, alors sa contrepartie en optimisation multi-objectif sera également NP-difficile. Même si le problème mono-objectif peut être résolu en un temps polynomial, il arrive que sa contrepartie en optimisation multi-objectif soit tout de même NP-difficile [27].

1.1.2 Solutions optimales et dominance

Dans ce manuscrit, on considère sans perte de généralisation vouloir minimiser les objectifs du problème d'optimisation. Les fonctions objectifs d'un problème multi-objectif sont généralement contradictoires, c'est-à-dire qu'il n'existe pas de solution unique optimisant tous les objectifs en même temps. Vient alors la question de comment comparer des solutions d'un problème multi-objectif. Par exemple, sur la Figure 1.1, la solution a est de meilleure qualité selon l'objectif f_2 mais de moins bonne qualité selon l'objectif f_1. Alors que la solution b est de bonne qualité selon l'objectif f_1 et de moins bonne qualité selon l'objectif f_2. Ces deux solutions sont donc incomparables.

Pour comparer des solutions dans un contexte d'optimisation multi-objectif, il est courant d'utiliser la relation de dominance de Pareto. Plus formellement, un vecteur objectif $z \in Z$ domine un autre vecteur objectif $z' \in Z$ si et seulement si $\forall i \in \{1,\dots,m\}$, $z_i \leq z'_i$ et $\exists j \in \{1,\dots,m\}$ tel que $z_j < z'_j$. On peut alors dire qu'une solution $x \in X$ est dominée par une solution $y \in X$ si $F(x)$ est dominé par $F(y)$. Mathématiquement, on note qu'une solution $x \in X$ domine une autre solution $x' \in X$ par la notation $x \succ x'$. Une solution $x \in X$ est Pareto optimale, ou non-dominée, si et seulement si $\forall x' \in X$, $x' \not\succ x$. L'ensemble des solutions Pareto optimales est appelé *l'ensemble Pareto* dans l'espace de décision. Son image dans l'espace objectif est appelée *le front Pareto*. L'ensemble Pareto peut être noté $X^\star = \left\{x \in X \mid \nexists x' \in X, x' \succ x\right\}$ et sa projection dans l'espace objectif est alors notée $Z^\star = \left\{f(x) \mid x \in X^\star\right\}$.

Grâce à ces définitions, on peut maintenant dire sur la Figure 1.1, que parmi les solutions $\{a,b,c,d,e\}$ les solutions $\{a,b,c\}$ sont des solutions non-dominées et les solutions $\{d,e\}$ sont dominées. En pratique, trouver une solution Pareto optimale est souvent NP-difficile, et la cardinalité de l'ensemble (ou du front) Pareto est souvent *intractable* [27]. Cette difficulté impacte directement les méthodes de résolutions des problèmes d'optimisation combinatoire multi-objectif, comme nous allons le voir dans la section suivante.

1.2 Résolution de problèmes d'optimisation multi-objectif

1.2.1 Approches de résolution

En optimisation, le décideur est la personne qui souhaite choisir une solution pour répondre au problème auquel il est confronté. La résolution d'un problème d'optimisation multi-objectif a pour but de présenter au décideur la solution qui correspond le mieux à ses préférences. Deux questions essentielles sont à se poser pour choisir la

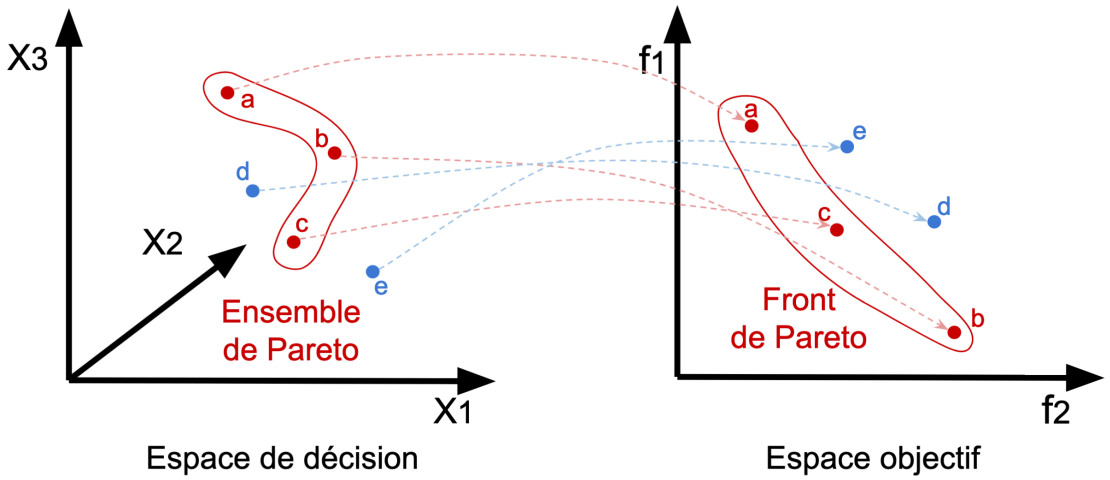

FIGURE 1.1 – Représentations de solutions dominées (en bleu) et non-dominées (en rouge) dans l'espace objectif et l'espace de décision.

méthode de résolution qui convient le mieux au contexte d'optimisation : Quelle est l'implication du décideur dans le processus de résolution ? Est-ce que le décideur peut se contenter d'une solution proche de la solution optimale ?

1.2.1.1 Définir le niveau d'implication du décideur

Le décideur peut être impliqué de différentes façons dans la résolution du problème d'optimisation multi-objectif. La méthode de résolution doit alors être choisie en fonction de cette implication, il existe ainsi trois niveaux d'implication :

— *a priori* : Dans ce cas, le décideur donne ses préférences *avant* le processus d'optimisation. La formulation et la modélisation des préférences du décideur peut être difficile selon la nature du problème. Il est par exemple courant d'utiliser les préférences du décideur pour modéliser un nouveau problème d'optimisation avec un objectif unique. Cette simplification permet alors d'utiliser une méthode d'optimisation mono-objectif. La difficulté dans cette approche réside dans la modélisation des préférences qui ne doit pas perdre les informations et les caractéristiques du problème multi-objectif initial.

— *a posteriori* : Le but de cette approche est de trouver (ou approximer) l'ensemble Pareto. Avec cette approche, le décideur intervient seulement à la fin du processus, où il doit choisir, parmi un ensemble de solutions retournées par le processus d'optimisation, la solution qui lui convient le mieux selon ses besoins et ses préférences. Le choix de la solution peut être un choix compliqué car l'ensemble

de solutions non-dominées trouvées par l'algorithme peut contenir énormément de solutions, notamment pour les problèmes avec beaucoup d'objectifs.

— *interactive* : Avec cette approche, le décideur est impliqué durant toute la phase de résolution. Il donne ses préférences pendant l'exécution de la méthode de résolution pour diriger itérativement le processus de recherche dans une direction qu'il aura choisi. Le but de cette approche est d'utiliser toutes les connaissances acquises sur le problème pour se diriger plus rapidement vers une solution répondant aux préférences et besoin du décideur. L'inconvénient est que le décideur doit être disponible pendant tout le processus d'optimisation.

L'approche utilisée sera choisie en fonction de la difficulté du problème et de l'implication du décideur. Dans le contexte de cette thèse, on se concentrera sur les méthodes *a posteriori*. On considérera ici la résolution d'un problème comme étant l'identification d'une approximation de l'ensemble Pareto. L'approximation de l'ensemble Pareto fait référence aux solutions obtenues par l'algorithme d'optimisation multi-objectif. En effet, comme nous l'avons vu précédemment, il est souvent impossible de trouver toutes les solutions non-dominées d'un problème multi-objectif. Dans ce cas, le but du processus de recherche est de trouver un ensemble de solutions proche de l'ensemble Pareto. En pratique, après avoir obtenu cette approximation, une phase d'aide à la décision permet d'aider le décideur à choisir une solution parmi l'ensemble des solutions non-dominées obtenues. Dans cette thèse, nous nous concentrons uniquement sur l'obtention d'une approximation de l'ensemble Pareto.

1.2.1.2 Méthode exacte ou méthode approchée ?

Parmi les approches *a posteriori*, deux grandes familles de méthodes de résolution existent : les méthodes exactes et les méthodes approchées.

Les méthodes de résolution exactes en optimisation multi-objectif permettent de garantir l'obtention de l'ensemble Pareto. Ces méthodes sont très coûteuses en ressource de calcul pour les problèmes difficiles. L'algorithme *Branch and Bround* [85] est un exemple de méthodes exactes existantes. Avec cet algorithme, on est assuré de trouver l'ensemble des solutions non-dominées sans avoir besoin d'énumérer et d'évaluer toutes les solutions de l'espace de décision. Les approches exactes ne sont cependant pas utilisables pour les problèmes *boite-noire*, pour lesquels on ne connaît pas la forme analytique des fonctions objectifs.

Par ailleurs, en optimisation combinatoire multi-objectif, les problèmes rencontrés sont souvent NP-difficiles, ce qui rend les méthodes exactes inutilisables en pratique

sur les instances de grande taille. Les méthodes de résolution approchées sont donc utilisées pour pallier les problèmes de performance des méthodes exactes. Les méthodes approchées permettent de trouver des solutions de bonne qualité en un temps de calcul raisonnable. Avec ces méthodes, nous n'avons aucune garantie sur l'obtention de l'ensemble Pareto, le but étant pour les instances les plus compliquées de trouver une approximation de cet ensemble. Les méthodes approchées doivent donc être choisies seulement si le décideur peut se contenter d'une approximation de l'ensemble Pareto. Étant donné que nous nous trouvons dans un contexte boite-noire et que nous considérons l'approximation de problèmes NP-difficiles, nous nous intéresserons dans cette thèse aux méthodes de résolution approchées, qui sont discutées ci-dessous.

1.2.1.3 Approches heuristiques

Les méta-heuristiques sont des méthodes de résolution algorithmiques qui permettent de trouver des solutions de bonne qualité en un temps de calcul raisonnable. Ces algorithmes sont constitués d'un ensemble de composants génériques qui définissent la manière d'explorer l'espace de recherche dans le but de trouver une bonne approximation de l'ensemble Pareto. Les méta-heuristiques peuvent s'inspirer de processus naturels comme les algorithmes d'essaims [49], le recuit simulé [51] ou les algorithmes évolutionnaires [22].

Il existe deux grandes classes de méta-heuristiques, les méta-heuristiques à base de solution unique et les méta-heuristiques à base de population. Les méta-heuristiques à base de solution unique ne gèrent qu'une solution à la fois durant le processus de recherche et retourne donc une solution unique à la fin du processus. Les algorithmes les plus connus de ce type sont la recherche locale et ses variantes, la recherche tabou [29], ou encore le recuit simulé [51]. À l'inverse, les méta-heuristiques à base de population manipulent un ensemble de solutions. Dans cette classe, nous pouvons trouver les algorithmes d'essaims [49], les algorithmes de colonies de fourmis [18], ou encore les algorithmes évolutionnaires [22], auxquels nous nous intéressons tout particulièrement dans le contexte de cette thèse. Les algorithmes évolutionnaires sont en effet bien adaptés aux problèmes d'optimisation multi-objectif, car ils sont naturellement conçus pour trouver un ensemble de solutions qui, en optimisation multi-objectif, correspond donc à une approximation de l'ensemble Pareto [22].

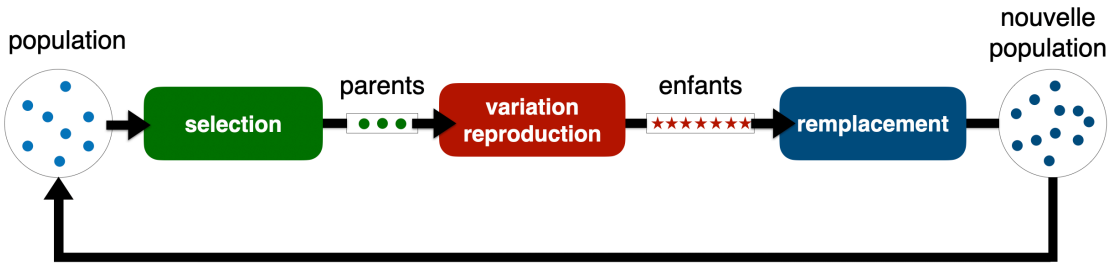

FIGURE 1.2 – Illustration d'un algorithme évolutionnaire.

1.2.2 Algorithmes évolutionnaires multi-objectifs

Les algorithmes évolutionnaires [22] sont des algorithmes à base de population qui ont prouvé leur robustesse et leur efficacité dans l'approximation de l'ensemble Pareto. Suite à l'initialisation de la population, ces algorithmes sont basés sur un processus générique et itératif expliqué ci-dessous et illustré dans la Figure 1.2 :

1. **Phase de sélection.** La première phase consiste à sélectionner un certain nombre de solutions, que l'on nommera *parents*, parmi les solutions présentes dans la population. Ces solutions seront utilisées dans la phase suivante pour générer de nouvelles solutions. Le choix des solutions *parents* est donc important, car les nouvelles solutions seront basées sur les caractéristiques de ces solutions. L'idée est d'améliorer la population en se basant sur les caractéristiques qui font que les solutions présentes dans la population soient de bonne qualité. On souhaite donc à cette étape choisir les meilleurs parents possibles pour la suite du processus.
2. **Phase de reproduction.** Cette deuxième phase consiste à générer de nouvelles solutions que l'on nommera *enfants*. Les solutions *enfants* sont générées à l'aide d'opérateurs stochastiques de variation. Deux types d'opérateurs sont étudiés dans ce manuscrit : le *croisement* qui échange les caractéristiques des deux solutions *parents* pour former une nouvelle solution et la *mutation* qui fait varier les caractéristiques d'une unique solution.
3. **Phase de remplacement** : Cette dernière phase consiste à remplacer une partie des solutions de la population par les nouvelles solutions *enfants*. L'objectif dans cette phase est d'améliorer la qualité de la population en y ajoutant les nouvelles solutions de bonne qualité, et en supprimant celles qui sont de moins bonne qualité. Il faut toutefois noter que même les solutions qui ne sont pas de très bonne qualité ont un rôle à jouer dans la population. En effet, la diversité des solutions dans les algorithmes évolutionnaires est importante, car elle permet au

processus de recherche d'explorer l'espace de recherche dans plusieurs régions potentiellement intéressantes, et qui n'auraient pas pu être explorées en ne gardant que des solutions d'excellente qualité.

En dehors du problème du choix des opérateurs de variation (qui dépend de la nature du problème), la conception de la phase de sélection des solutions parents (étape 1) et de l'opérateur de remplacement (étape 3) est vu comme le principal défi pour obtenir un algorithme évolutionnaire multi-objectif efficace. En effet, ces étapes permettent de contrôler la convergence de la population vers l'ensemble Pareto, ainsi que sa diversité.

Ainsi, il existe différents types d'algorithmes évolutionnaires multi-objectif, qui se distinguent dans la manière de faire évoluer la population de solutions, c'est-à-dire, dans la manière de concevoir les étapes de sélection et de remplacement. Les approches basées sur la dominance, sur les indicateurs, et sur la décomposition sont présentées ci-dessous.

1.2.2.1 Approches basées sur la dominance

Les approches basées sur la dominance, comme NSGA-II [22] ou MOGA [74], s'appuient sur la relation de dominance entre les solutions dans leurs étapes de sélection et de remplacement pour faire évoluer la population de l'algorithme.

NSGA-II est souvent considéré comme l'algorithme état-de-l'art des approches basées sur la dominance. Il se différencie au niveau de son processus de remplacement, qui est illustrée dans la Figure 1.3. Ce processus commence par combiner l'ensemble des solutions présentes dans l'algorithme, c'est-à-dire les solutions *parents* et les solutions *enfants* qui ont été générée à l'itération courante, avant ce processus de remplacement. Les solutions sont triées en fonction d'un rang de dominance, c'est-à-dire que les solutions non-dominées auront un rang de 1 et correspondent au front $\mathcal{F}^1$, le front suivant $\mathcal{F}^2$ correspond aux solutions non-dominées exceptées les solutions du front $\mathcal{F}^1$, et ainsi de suite pour les fronts suivants. Si la taille de la population le permet, toutes les solutions ayant les rangs les plus faibles sont conservées dans la nouvelle population. En cas d'égalité de rang, il ne faut conserver qu'un sous-ensemble des solutions du même rang, comme on peut le voir avec le rang 3 dans la Figure 1.3. Dans ce cas, la distance de *crowding* est utilisée et seules les solutions les plus espacées dans l'espace objectif sont conservées. La distance de crowding est une mesure de dispersion, elle correspond à la distance moyenne entre deux solutions dans l'espace objectif. Plus la distance entre une solution et ses solutions voisines sera grande, et plus cette valeur sera importante, et donc plus la solution aura de chance de survie.

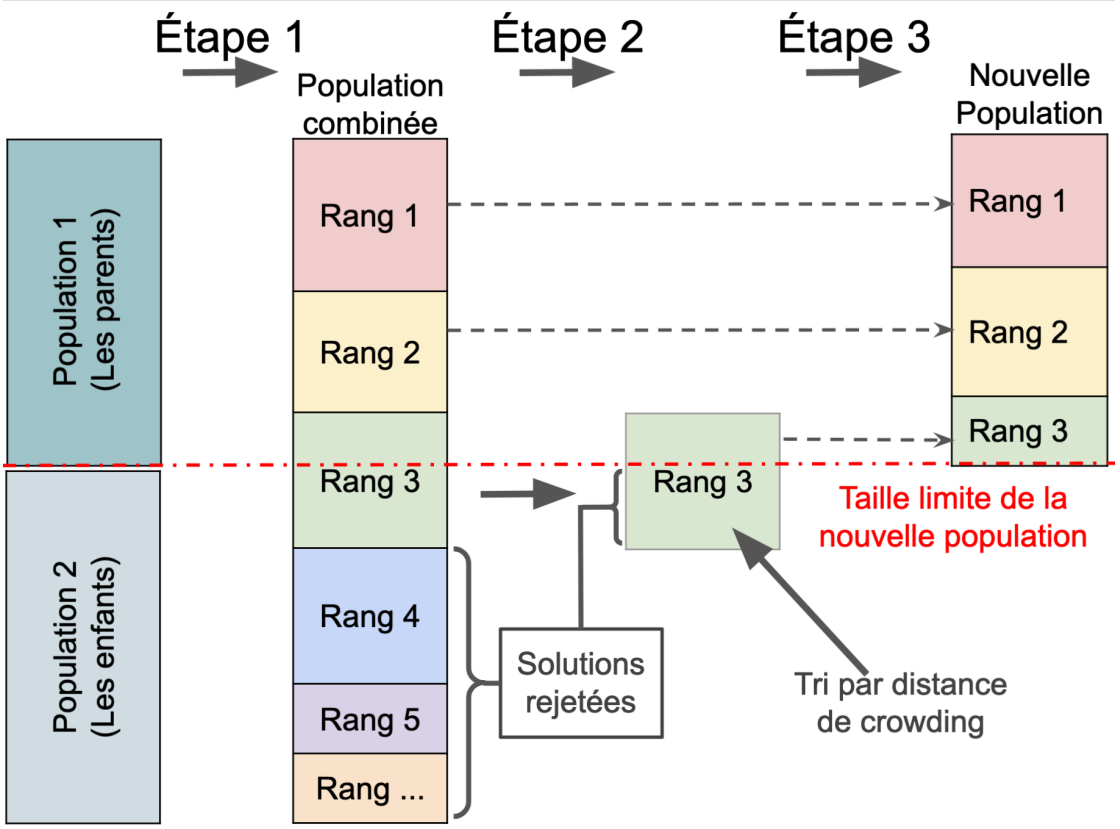

FIGURE 1.3 – Processus de remplacement dans NSGA-II.

1.2.2.2 Approches basées sur les indicateurs

Les approches basées sur les indicateurs, tels que IBEA [121], MaOEA/IGD [94] ou encore SMS-EMOA [8], s'appuient sur un indicateur de performance pour guider le processus de recherche. Les indicateurs de performance [61] sont utilisés pour évaluer la qualité de l'approximation du front Pareto trouvée. Nous en présenterons en détail plus loin dans la Section 1.5. Dans ce type d'approches, on considère optimiser le problème d'optimisation multi-objectif en optimisant l'indicateur de performance de la population courante. Par exemple dans SMS-EMOA et IBEA, le but est de supprimer à chaque génération la ou les solutions de la population qui contribuent le moins à l'indicateur de performance.

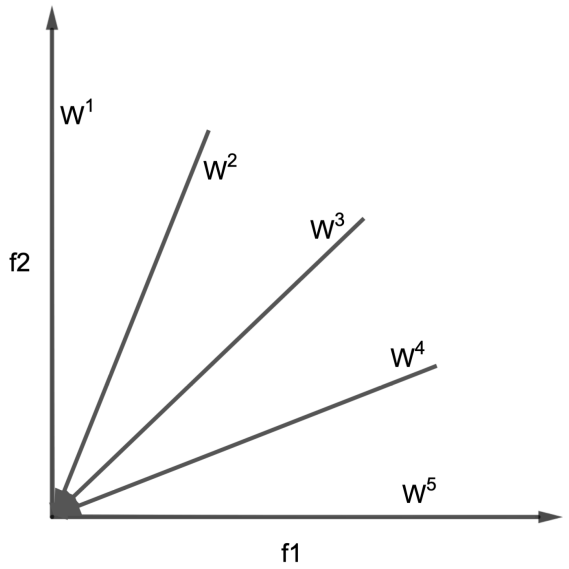

FIGURE 1.4 – Représentation de 5 vecteurs de poids $W = \left\{w^1, w^2, w^3, w^4, w^5\right\}$ pour un problème à 2 objectifs.

1.2.2.3 Approches basées sur la décomposition

La décomposition d'un problème multi-objectif consiste à le découper en plusieurs sous-problèmes mono-objectifs. Un sous-problème est généralement défini par une fonction d'agrégation et par un vecteur de poids donnant une direction de recherche au sous-problème dans l'espace objectif. On parle de direction de recherche car le vecteur de poids permet de pondérer chacun des objectifs et de moduler l'importance que portera le sous-problème sur les différents objectifs. Ceci est illustré dans la Figure 1.4. Sur cette figure, le vecteur de poids w^2 donne plus d'importance à l'objectif f_2 alors que le vecteur de poids w^4 donnera lui plus d'importance à l'objectif f_2. Pour couvrir l'ensemble du front Pareto, il est donc évident que la répartition des vecteurs de poids dans l'espace objectif est un élément important.

Une approche basée sur la décomposition permet d'utiliser des techniques issues de l'optimisation mono-objectif et de simplifier la comparaison entre les solutions en se mettant au niveau des sous-problèmes mono-objectifs. Ce type d'approches est l'objet de cette thèse et sera présentée plus en détail dans la section suivante, en particulier l'algorithme état-de-l'art MOEA/D [87].

Algorithme 1 : MOEA/D

Input : $\mathcal{W} := \left\{w^1, \ldots, w^\mu\right\}$: ensemble de vecteurs de poids ; $g(\cdot \mid \omega)$: fonction d'agrégation ; T : taille du voisinage ;

$\mathcal{EP} \leftarrow \varnothing$: (Facultatif) archive externe ;

$\mathcal{P} \leftarrow \left\{x^1, \ldots, x^\mu\right\}$: générer et évaluer la population initiale de taille μ;

$z^\star \leftarrow$ initialiser le point de référence avec la population initiale $\mathcal{P}$;

$\mathcal{B} \leftarrow$ calcul du voisinage de chaque sous-problème i avec les T plus proches sous-problèmes;

tant que *le critère d'arrêt n'est pas atteint* **faire**

    /* Parcours du sous-problème i */

    pour $i \in \{1, \ldots, \mu\}$ **faire**

        /* Sélection des parents dans le voisinage $\mathcal{B}_i$ */

        $\mathcal{X} \leftarrow \text{parents}(\mathcal{B}_i)$;

        /* Utilisation d'opérateurs de variation pour générer une solution enfant à partir de $\mathcal{X}$ */

        $x' \leftarrow \text{reproduction}(\mathcal{X})$;

        /* Evaluation de la solution enfant x' */

        $F(x') \leftarrow \text{evaluation}(x')$;

        /* Mise à jour du point de référence avec $F(x')$ */

        $z^\star \leftarrow \text{mise\_a\_jour\_z}(z^\star, F(x'))$;

        /* mise à jour de l'archive externe des solutions non-dominées avec la nouvelle solution x' */

        $\mathcal{EP} \leftarrow \text{mise\_a\_jour\_archive}(\mathcal{EP}, x')$;

        /* processus de remplacement de la solution x' dans le voisinage $\mathcal{B}_i$ */

        pour $j \in \mathcal{B}_i$ **faire**

            si $g(F(x')|w^j)$ ***est meilleur que*** $g(F(x^j)|w^j)$ **alors**

                $\mathcal{P}_j \leftarrow x'$;

1.3 Algorithmes évolutionnaires multi-objectifs par décomposition

L'algorithme MOEA/D [87] est le framework état-de-l'art pour les algorithmes évolutionnaires multi-objectifs basés sur la décomposition. Il s'est imposé grâce à ses très bonnes performances dans la découverte de solutions dans de multiples régions du front Pareto, à sa simplicité, et à son faible coût computationnel [100, 107]. Il attire notamment par ses avantages liés à la décomposition, mais aussi à sa parallélisation possible pour augmenter les performances de l'algorithme [7]. Par ailleurs, MOEA/D est largement

utilisé pour résoudre des problèmes d'ingénierie complexes. Il est utilisé par exemple pour optimiser la structure d'un réseau de neurones pour l'apprentissage profond [63], pour des problèmes d'ordonnancement [43, 42, 15], ou encore pour des problèmes de réduction de coût [116], pour citer quelques applications remarquables. Le framework est détaillé dans l'Algorithme 1. Chaque composant est expliqué dans les sous-sections suivantes, en faisant référence aux lignes correspondantes de l'Algorithme 1.

1.3.1 La décomposition dans MOEA/D

L'algorithme décompose le problème multi-objectif en plusieurs sous-problèmes mono-objectifs, grâce à une fonction d'agrégation g et à un ensemble de vecteur de poids $\mathcal{W} := \left\{w^1, \ldots, w^\mu\right\}$. Chaque sous-problème mono-objectif i est ainsi défini grâce à la combinaison de la fonction d'agrégation g et d'un vecteur de poids w^i. La conception de MOEA/D suit les fondations d'un algorithme évolutionnaire, où l'on retrouve les phases de sélection, de reproduction et de remplacement. La population $\mathcal{P}$ de taille μ (ligne 2) est organisée en attribuant chacune de ses solutions à un vecteur de poids différent. Ainsi, la solution $x^i \in \mathcal{P}$ est liée au vecteur de poids $w^i \in \mathcal{W}$ correspondant au sous-problème i. La taille de la population est donc la même que le nombre de vecteurs de poids. Au fur et à mesure des itérations, les sous-problèmes sont optimisés de manière coopérative pour améliorer les solutions de la population. Les deux éléments clés dans la décomposition en sous-problèmes sont les vecteurs de poids et la fonction d'agrégation qui sont expliqués plus en détail ci-dessous.

1.3.1.1 Vecteurs de poids

La taille de la population μ représente à la fois le nombre de solutions dans la population, mais également le nombre de vecteurs de poids utilisés par MOEA/D, car chaque solution de la population est liée à un vecteur de poids. La configuration de ce paramètre est importante et difficile, car il correspond à l'ensemble de solutions retournées par l'algorithme, et donc à l'approximation de l'ensemble Pareto. Une mauvaise configuration de ce paramètre a des effets soit sur la qualité du front obtenue par l'algorithme, soit sur le temps de calcul. En effet, une valeur trop petite limite le nombre de sous-problèmes et ne permet pas à l'algorithme d'approcher la totalité du front Pareto. À l'inverse, un nombre de solutions trop important dans la population peut réduire les performances de l'algorithme en perdant du temps et du budget. Si le nombre de sous-problèmes est trop important, les coefficients des vecteurs de poids sont trop proches et la probabilité que les sous-problèmes voisins aient la même solution optimale est de plus en plus

grande.

Une fois la taille de la population fixée, et donc le nombre de vecteurs de poids, il reste encore à déterminer la façon de les générer avant l'exécution de l'algorithme. La génération des vecteurs de poids est un véritable défi quand le nombre d'objectifs augmente. Il existe différentes approches pour les générer, qui seront choisi en fonction du contexte d'optimisation dans lequel l'algorithme se trouve [107]. L'approche la plus utilisée pour générer les vecteurs de poids est la génération uniforme. La conception *Simplex-lattice* [91] permet une distribution des vecteurs la plus uniforme possible, mais devient plus limitée quand le nombre d'objectifs augmente, en perdant en uniformité [115]. Les conceptions *axial* [19] et *simplex-centroid* [90] souffrent du même problème. La conception *Simplex-lattice à double couche* [21] permet de pallier ce problème en générant deux couches de vecteurs de poids. D'autres stratégies ont également été proposées pour améliorer la génération et la répartition des vecteurs de poids [95] comme la méthodologie SOBOL [115] que nous utilisons dans ce manuscrit.

Dans certain cas, il peut être intéressant d'adapter les vecteurs de poids au cours de l'algorithme, en fonction de la forme du front Pareto. En effet, quand le front est irrégulier, un ensemble de vecteurs de poids uniforme n'est pas optimal, en particulier pour les fronts discontinus où plusieurs sous-problèmes auront la même solution optimale. Les stratégies AWA [86] ou DMEA-WUA [64] permettent ainsi d'orienter la recherche vers des zones utiles, en évitant les parties discontinues du front. Dans la même idée d'adapter les poids en fonction de la forme du front, il existe une méthode adaptative pour ajuster les vecteurs de poids en fonction des caractéristiques géométriques du front, notamment lorsque le front est concave [92].

Par ailleurs, il se peut que la difficulté d'optimisation des sous-problèmes soit inégale [117]. Cette inégalité est également un facteur d'ajustement au niveau des directions de recherche, afin de donner plus de budget aux sous-problèmes les plus difficiles. Ces adaptations peuvent avoir lieu soit directement dans MOEA/D au moment du choix des sous-problèmes à parcourir sans adapter les vecteurs de poids, comme on le verra dans le chapitre suivant, soit directement au niveau des vecteurs de poids en adaptant leurs coefficients [33]. Le but de cette approche est de ne pas perdre de ressources de calcul sur des sous-problèmes peu utiles [60], ou de diviser les sous-problèmes compliqués pour allouer plus de temps de calcul dans certaines directions de recherche [33].

1.3.1.2 Fonctions d'agrégation

Une fonction d'agrégation, également appelée fonction scalaire, permet d'agréger plusieurs valeurs issues des fonctions objectifs en une seule valeur scalaire que l'on appellera valeur d'agrégation. Comme nous l'avons expliqué précédemment, le vecteur de poids utilisé dans le calcul de la valeur d'agrégation a un impact important. À l'origine de MOEA/D, trois fonctions d'agrégation étaient utilisées (la somme pondérée, Penalty Boundary Intersection, et Chebyshev-pondérée) [87] avant de voir l'arrivée de nouvelles fonctions et variantes pour améliorer ce composant, comme nous l'expliquons dans la suite de cette section. Nous présentons ci-dessous les deux fonctions les plus utilisées, avant de présenter les derniers travaux sur les fonctions d'agrégations dans MOEA/D.

La somme pondérée. La somme pondérée consiste à faire la somme du produit du vecteur de poids ω avec le vecteur objectif $F = \{f_1, ..., f_m\}$ où m représente le nombre d'objectifs. Nous souhaitons minimiser cette fonction si les objectifs du problème sont à minimiser, et inversement.

$$g^{ws}(x, \omega) = \sum_{i=1}^{m} \omega_i \cdot f_i \tag{1.2}$$

Néanmoins, certaines solutions de l'ensemble Pareto ne sont optimales pour aucune spécification du vecteur de poids. Ces solutions se trouvent hors de l'enveloppe convexe du front Pareto, et ne peuvent donc pas être trouvées en utilisant la somme pondérée comme fonction d'agrégation [100].

Chebyshev-pondérée. La fonction Chebyshev-pondérée est une des fonctions d'agrégation les plus utilisées dans la littérature. Cette fonction nécessite un point de référence $z^\star$. Le point de référence est un vecteur que l'on note $z^* = (z_1^*, \ldots, z_m^*)$, où m représente le nombre d'objectifs. Le point de référence est le point parfait qui optimise indépendamment chacun des m objectifs du problème. Chaque coordonnée z_i^* du point idéal z^* est représenté par :

$$z_i^* = \min_{x \in X} f_i(x),\ i \in \{1, \ldots, m\} \tag{1.3}$$

En considérant que le point de référence et les vecteurs de poids soient correctement choisis, cette approche donne la possibilité de trouver toutes les solutions non-dominées du front Pareto, même si le front est non-convexe [70]. Cette fonction consiste à minimiser la plus grande distance pondérée par un vecteur de poids $\omega = (\omega_1, \ldots, \omega_m)$ entre le

point de référence $z^\star$ et n'importe quelle solution $x \in X$.

$$g^{\text{chebyshev}}(x, \omega) = \max_{i \in \{1,\dots,m\}} \omega_i \cdot \left| z_i^\star - f_i(x) \right| \tag{1.4}$$

La fonction d'agrégation doit être choisie en fonction du problème considéré. Comme expliqué précédemment, la fonction d'agrégation *Somme Pondérée* n'est pas adaptée aux fronts Pareto concaves. De nombreux travaux de recherche proposent de nouvelles approches de décomposition utilisant différents types d'agrégation. Certain de ces travaux porte sur l'amélioration des fonctions existantes comme p–TCH [67], qui améliore la fonction Chebyshev en ajoutant des contraintes sur les vecteurs de poids pour mieux utiliser les propriétés géométriques de certains vecteurs de poids, ou encore [101] qui ajoute des contraintes sur les sous-problèmes pour améliorer la convergence vers le front Pareto ainsi que la diversité de la population. De nouvelles fonctions ont également été proposées [13] (*constrained decomposition with grids*). Elles visent à décomposer l'espace objectif non plus grâce à des vecteurs de poids, mais à l'aide de grilles en vue d'améliorer la robustesse de l'algorithme même avec des fronts Pareto compliqués. Jiang et al. proposent également deux nouvelles fonctions paramétrables pour contrôler les régions d'amélioration de chaque sous-problème [44]. Certains travaux combinent plusieurs fonctions d'agrégation [40, 118], où les fonctions somme-pondérée et Chebyshev sont utilisées en coopération. Il a d'ailleurs été proposé une méthode adaptative pour choisir au cours de l'algorithme la meilleure fonction à utiliser en fonction de la qualité de l'approximation obtenue [39].

Selon une étude récente sur les différentes approches de décomposition [100], la fonction d'agrégation doit être choisie en fonction du problème à optimiser. Dans un contexte boite-noire où la forme analytique des objectifs à optimiser n'est pas connue, le choix de la fonction d'agrégation devient un vrai défi car on ne connaît pas la forme du front Pareto. L'étude montre alors que la fonction Chebyshev reste un bon compromis avec des bons résultats sur l'ensemble des problèmes considérés par les auteurs de ces travaux et semble donc recommandée pour les problèmes boite-noire [100].

1.3.2 Voisinage

Une des caractéristiques de MOEA/D est la coopération entre les sous-problèmes durant le processus de recherche. En effet, l'algorithme optimise les différents sous-problèmes de manière coopérative, en les faisant interagir avec leurs sous-problèmes voisins. Le système de voisinage est un mécanisme qui permet non seulement à un

sous-problème de bénéficier des solutions courantes de ses voisins, mais aussi de potentiellement améliorer ses voisins par le biais de solutions nouvellement créées. La relation de voisinage est définie en utilisant la distance entre les vecteurs de poids et la taille du voisinage définie par le paramètre T. En utilisant cette relation de voisinage, on considère que les vecteurs de poids voisins sont similaires, et donc que les solutions liées à ces sous-problèmes voisins sont également similaires. Chaque sous-problème est optimisé grâce aux informations accessibles dans le voisinage. L'idée est d'utiliser toutes les informations qui font qu'une solution est de bonne qualité pour un sous-problème donné, pour optimiser plus vite ce sous-problème et ses voisins. Le système de voisinage est donc utilisé dans un premier temps au moment de la sélection des solutions parents $\mathcal{X}$ (ligne 7), pour permettre d'avoir un ensemble de solutions assez proches pour générer une solution enfant x'. On l'utilise ensuite dans un second temps durant le processus de remplacement (ligne 12), il permet ainsi un remplacement local des solutions liées aux sous-problèmes voisins.

MOEA/D–DE [59] est une des premières variantes qui modifie le système de voisinage proposé par défaut dans MOEA/D. Cette variante ajoute un paramètre δ qui désigne la probabilité de choisir soit le voisinage classique $\mathcal{B}_i$, soit toute la population $\mathcal{P}$, à la fois pour la phase de sélection des parents (ligne 7 de l'Algorithme 1) et pour la phase de remplacement (ligne 12). Dans la version originale de MOEA/D, la sélection des solutions parents et le processus de remplacement utilisent toujours le voisinage $\mathcal{B}_i$. L'ajout de ce nouveau mécanisme dans MOEA/D–DE permet de choisir avec une probabilité non-nulle des parents plus éloignés de la solution courante x^i liée au sous-problème i. Ce nouveau mécanisme permet également de remplacer des solutions de la population qui seraient améliorées par la nouvelle solution x', même si ces solutions sont en dehors du voisinage $\mathcal{B}_i$. En diminuant la probabilité δ, on permet à l'algorithme d'explorer plus intensément l'espace de recherche en évitant les limitations d'un voisinage restreint. La valeur habituellement recommandée est une probabilité de 0.9, ainsi le voisinage sera utilisé en moyenne 90% du temps et donc la totalité de la population les 10% restants [59, 117, 17, 47].

1.3.3 Génération des solutions enfants

MOEA/D utilise un processus évolutionnaire pour générer de nouvelles solutions dans son processus de recherche. Les composants évolutionnaires de MOEA/D sont dépendants des mécanismes internes à l'algorithme comme le système de voisinage. En effet, on a pu voir précédemment que le processus de sélection des solutions parents est

intimement lié aux solutions présentes dans le voisinage $\mathcal{B}_i$ du sous-problème i (ligne 7). L'utilisation des solutions voisines $\mathcal{B}_i$ comme solutions parents permet d'utiliser les caractéristiques qui font que ces solutions soient bonnes pour le sous-problème i. À chaque parcours d'un sous-problème i, une nouvelle solution x' est générée (ligne 8) grâce à des opérateurs de variation (croisement, mutation). Elle est ensuite évaluée (ligne 9) pour connaître ses valeurs objectifs.

1.3.4 Mise à jour de la population

MOEA/D est un algorithme évolutionnaire avec deux populations, $\mathcal{P}$ et $\mathcal{EP}$. La population $\mathcal{EP}$ (ligne 1) est une archive externe facultative qui a pour rôle d'archiver toutes les solutions non-dominées rencontrées tout au long du processus de recherche. L'archive externe $\mathcal{EP}$ ne joue pas de rôle dans le processus de recherche, elle a un simple rôle d'archive des solutions non-dominées pour stocker l'approximation de l'ensemble Pareto. L'archive $\mathcal{EP}$ devient souvent de plus en plus grande au fur et à mesure des itérations, et ce phénomène est accentué quand le nombre d'objectifs est important.

Grâce à l'évaluation de chaque solution générée x', cette nouvelle solution est ajoutée à l'archive externe $\mathcal{EP}$ (ligne 11) si elle n'est pas dominée par une autre solution de l'archive $\mathcal{EP}$. Après cet ajout, les solutions dominées par x' dans l'archive sont supprimées. Par ailleurs, le point de référence $z^\star$ est mis à jour si une des coordonnées de $F(x')$ améliore celles de $z^\star$ dans l'espace objectif (ligne 10).

Enfin, la nouvelle solution enfant x' va également pouvoir remplacer une ou plusieurs solutions au sein de la population principale $\mathcal{P}$ (ligne 12). Comme nous l'avons vu précédemment, ce remplacement est local et ne peut se faire que dans le voisinage $\mathcal{B}_i$ (sans considérer le paramètre δ issu de MOEA/D–DE). La solution x' remplacera les solutions $x^k \in \mathcal{P}$ tel que $k \in \mathcal{B}_i$ si et seulement si la valeur d'agrégation $g(x' \mid w^k)$ est meilleure que la valeur d'agrégation $g(x^k \mid w^k)$.

Depuis la première version de MOEA/D en 2007, plusieurs variantes ont été proposées pour améliorer le système de remplacement de MOEA/D. MOEA/D–DE [59], via le paramètre nr, permet de garder un contrôle sur la diversité de la population en limitant le nombre de remplacements. Ce paramètre s'attaque au problème de convergence prématurée qui est bien connu des algorithmes évolutionnaires [109]. On peut en effet se retrouver dans certains cas avec une nouvelle solution enfant x' de très bonne qualité selon tous les objectifs, et qui remplace énormément de solutions dans la population $\mathcal{P}$, ce qui réduit considérablement la diversité. L'impact est négligeable à court terme car on augmente la qualité globale de la population, mais à plus long terme cela va ralentir

le processus de recherche car les solutions générées seront de plus en plus identiques, ce qui fera perdre des évaluations inutiles. Ce nouveau paramètre *nr* permet ainsi de limiter le nombre de remplacements maximal autorisé pour garder une bonne diversité dans la population.

MOEA/D–STM [47] est également une variante qui modifie le système de remplacement des solutions dans la population. En effet, MOEA/D–STM propose de modifier le système de remplacement classique par un algorithme de jumelage stable (*Stable Matching*) entre les solutions et les sous-problèmes. Le but est d'allouer à chaque sous-problème la solution qui lui convient le mieux parmi toutes les solutions de la population ainsi que la nouvelle solution enfant *x'* générée ligne 8 de l'Algorithme 1.

1.4 Implémentation modulaire du framework MOEA/D

Comme nous l'avons montré dans la section précédente, l'algorithme MOEA/D peut être vu comme un framework modulaire. En effet, de nombreuses variantes ont été proposées au cours des dernières années. Chacune d'entre-elles ne modifient qu'une partie de l'algorithme, que l'on peut considérer comme un composant. Le fait de décomposer MOEA/D nous permet d'identifier et d'analyser plus facilement le comportement de chacune des phases importantes qui composent l'algorithme. Dans cette section, nous présentons les implémentations existantes de MOEA/D dans différents langages de programmation avant de présenter notre contribution modulaire de MOEA/D au sein d'une bibliothèque logicielle développée au cours de la thèse [81].

1.4.1 Les implémentations existantes de MOEA/D

La première version de MOEA/D ainsi que ces variantes les plus connus comme MOEA/D–DE et MOEA/D–DRA ont été intégrées aux logiciels d'optimisation multi-objectif les plus utilisés comme le montre le Tableau 1.1 qui en référence quelques-uns. Pymoo [10], jMetal [75] et Metaheuristics.jl [68] sont des bibliothèques développées respectivement en Python, Java et Julia. Ces bibliothèques sont assez générales et implémentent de nombreuses méta-heuristiques et outils pour l'optimisation multi-objectif. Néanmoins, elles ne permettent pas d'expérimenter en détail le comportement des différents composants des méta-heuristiques implémentées, ou de modifier en profondeur les éléments qui composent ces méta-heuristiques.

Par contraste, la bibliothèque MOEADr [14], développée en R, propose une implémentation modulaire de MOEA/D sous la forme de 8 composants paramétrables. Cette

Tableau 1.1 – Différentes implémentations de MOEA/D.

Package	Langage	Auteur / Éditeur	ref.
Pagmo / Pygmo	C++ /Python	Francesco Biscani et Dario Izzo	[9]
Pymoo	Python	Julian Blank et Kalyanmoy Deb	[10]
jMetal	Java	Antonio J. Nebro et al	[75]
Metaheuristics.jl	Julia	Jesus Mejía	[68]
MOEADr	R	Felipe Campelo et al	[14]

implémentation modulaire permet d'expérimenter facilement les différents composants de MOEA/D sans donner à l'utilisateur la possibilité d'en ajouter de nouveaux.

1.4.2 Le framework modulaire : moead-framework

Pour faciliter nos différents travaux durant cette thèse, nous avons développé un framework modulaire de MOEA/D en Python appelé *moead-framework* [81] qui est disponible sur GitHub [1]. Avec ce framework, nous apportons la modularité de la bibliothèque MOEADr en utilisant la flexibilité du langage Python. Le framework permet ainsi à l'utilisateur de modifier le comportement des différents composants de MOEA/D sans être limité par les composants préexistants.

La bibliothèque est basée sur une architecture modulaire permettant de facilement ajouter, modifier ou tester les composants de MOEA/D. Chaque composant est configurable, tout comme ses interactions avec les autres composants. En contraste avec les implémentations existantes de MOEA/D, *moead-framework* ne limite pas l'utilisateur avec un nombre prédéfini de composants disponibles comme le montre la Figure 1.5. Cette figure montre une représentation des composants disponibles dans le framework. On peut notamment y voir que le composant principal *Algorithm* permet de gérer les interactions entre les différents composants. Les utilisateurs pourront restructurer les 10 composants existants présentés et décrits dans le Tableau 1.2, mais aussi en ajouter facilement des nouveaux pour expérimenter de nouvelles fonctionnalités sans pour autant devoir modifier le comportement des autres composants. Parmi les 10 composants présents, l'utilisateur pourra par exemple configurer la fonction d'agrégation, le critère d'arrêt, ou encore la manière de générer de nouvelles solutions.

Le framework implémente actuellement trois versions majeures d'algorithme évolutionnaire multi-objectif basés sur la décomposition, qui sont MOEA/D [87], MOEA/D–DE [59], MOEA/D–DRA [117], ainsi que les différentes approches développées durant

1. https ://github.com/moead-framework/framework

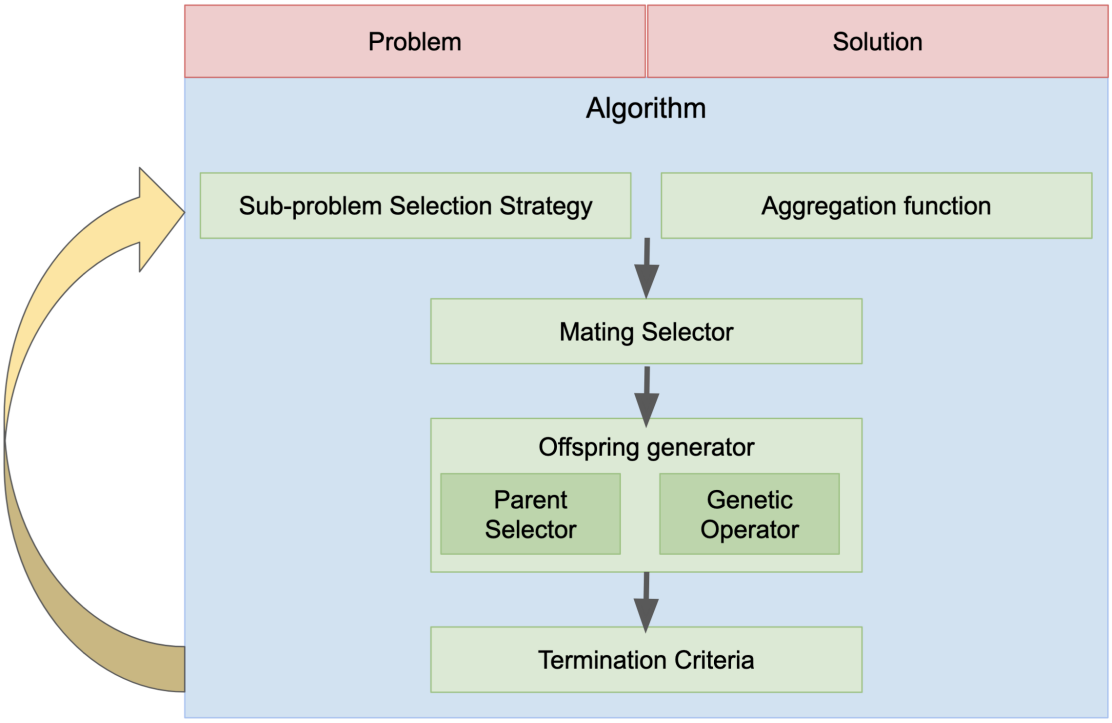

FIGURE 1.5 – Représentation des composants de la bibliothèque Python moead-framework.

la thèse. Il est ainsi possible de se baser sur ces implémentations par héritage pour implémenter sa propre version de MOEA/D ou alors configurer les composants passés en paramètres de l'algorithme.

1.5 Analyse de performance et problèmes étudiés

Il existe plusieurs moyens pour comparer et analyser les performances d'algorithmes d'optimisation combinatoires multi-objectifs. Pour cela, il s'agit d'évaluer et de comparer les approximations du front Pareto par le biais d'indicateurs de qualité que l'on pourra associer à des tests statistiques pour valider les résultats. Les algorithmes étudiés dans ce manuscrit sont stochastiques, ce qui implique que chaque algorithme doit être exécuté plusieurs fois pour permettre une analyse plus précise.

Dans cette thèse, on ne s'intéresse pas à la résolution d'un problème d'optimisation en particulier. À l'inverse, on s'intéresse à la compréhension des approches de décomposition en fonction des caractéristiques générales du problème et du scénario de résolution, que ce soit en termes de budget disponible, de temps de calcul, ou encore de

Tableau 1.2 – Description des composants de moead-framework.

Composant	Description
Problem	Permet de définir le problème d'optimisation
Solution	Représentation d'une solution (vecteur objectif et vecteur de décision)
Algorithm	Composant principale qui définit les interactions entre les différents composants
Sub-problem Selection Strategy	Permet de choisir les sous-problèmes itérés pendant la prochaine génération
Aggregation function	Permet de définir la fonction scalaire et la manière de comparer deux solutions
Mating Selector	Permet de sélectionner les solutions éligibles à être parents
Offspring generator	Permet de générer des solutions enfants en utilisant des sous-composants spécifiques
Parent Selector	Sélectionne les solutions parents parmi un ensemble de solution
Genetic Operator	Opérateur génétique permettant de générer des solutions à partir des solutions parents
Termination Criteria	Définit le critère d'arrêt de l'algorithme

qualité d'approximation escomptée. Dans cette section, nous présenterons les méthodes d'analyse de résultats et les problèmes que nous considérons dans le but d'étudier nos méthodes d'optimisation.

1.5.1 Indicateurs de qualité

En optimisation multi-objectif, les indicateurs de qualité permettent de donner une valeur scalaire à un ensemble de solutions obtenues par un algorithme. Cette valeur permet de nous aider à définir la qualité de l'approximation obtenue. L'obligation de mesurer à la fois la diversité des solutions et leur convergence vers le front Pareto rend la mesure de qualité de ces solutions très difficile. Plusieurs indicateurs de qualité ont été proposés ces dernières années [121, 122] pour calculer la qualité d'un ensemble de solutions. Dans cette thèse, nous utilisons deux indicateurs état-de-l'art différents, qui ont pour but de mesurer à la fois la convergence et la diversité : l'hypervolume et l'indicateur epsilon présentés ci-dessous. Ces deux indicateurs peuvent être utilisés directement dans certains types d'algorithmes pour guider le processus de recherche dans le but d'améliorer la valeur de l'indicateur de qualité, comme nous l'avons vu

précédemment. Dans cette thèse, nous les utilisons uniquement pour comparer et analyser la performance des différentes configurations de nos algorithmes.

Ces différentes mesures de qualité peuvent être utilisées pour montrer l'évolution de la qualité des solutions obtenues au fur et à mesure de l'exécution, c'est ce que l'on appelle étudier le profil de convergence [24]. Le profil de convergence sert à étudier le comportement de l'algorithme au fur et à mesure du temps. On appelle ainsi comportement "anytime" le fait de considérer le profil de convergence d'un algorithme, sans se soucier de la condition d'arrêt de l'algorithme. L'étude du comportement anytime d'un algorithme est très important en contexte boîte-noire, car on ne connaît pas à l'avance le budget qui sera alloué au processus d'optimisation.

1.5.1.1 Indicateur hypervolume

L'indicateur hypervolume [120] mesure le volume de la portion de l'espace objectif qui est dominée par l'approximation du front Pareto. Cet indicateur est dépendant d'un point de référence noté z^{REF}. Ce point doit être dominé par toutes les solutions considérées dans l'approximation du front Pareto. La définition de ce point est importante et peut jouer un rôle sur la valeur obtenue. Par convention, on définit les coordonnées de z^{REF} par les pires valeurs obtenues pour chacun des objectifs parmi toutes les solutions identifiées, sans distinction de l'algorithme qui a pu les trouver. Nous cherchons à maximiser l'hypervolume car plus sa valeur est grande et plus l'approximation du front Pareto est de bonne qualité. La Figure 1.6 représente l'hypervolume par la zone colorée. Plus cette zone est importante, plus la valeur de l'hypervolume, et par conséquent la qualité d'approximation du front, est élevée pour un même point z^{REF}.

Afin de normaliser les valeurs de l'hypervolume, nous considérons *la déviation relative de l'hypervolume*, noté I_{HVRD}. On va ainsi normaliser l'hypervolume de l'approximation par l'hypervolume calculé sur un front de référence, qui correspond au meilleur front connu, sans distinction de l'algorithme étudié :

$$I_{HVRD}(A) = \frac{I_H(R) - I_H(A)}{I_H(R)} \tag{1.5}$$

où A représente l'approximation du front Pareto que l'on étudie et R représente le front de référence. Nous construisons le front de référence pour un problème donné en combinant toutes les solutions retournées par les algorithmes considérés, et en filtrant les solutions non-dominées. Plus la valeur de $I_{HVRD}(A)$ est petite et plus la qualité de A est bonne. On peut également noter que $I_{HVRD}(A) = 0$ signifie que l'approximation étudiée est identique au front de référence.

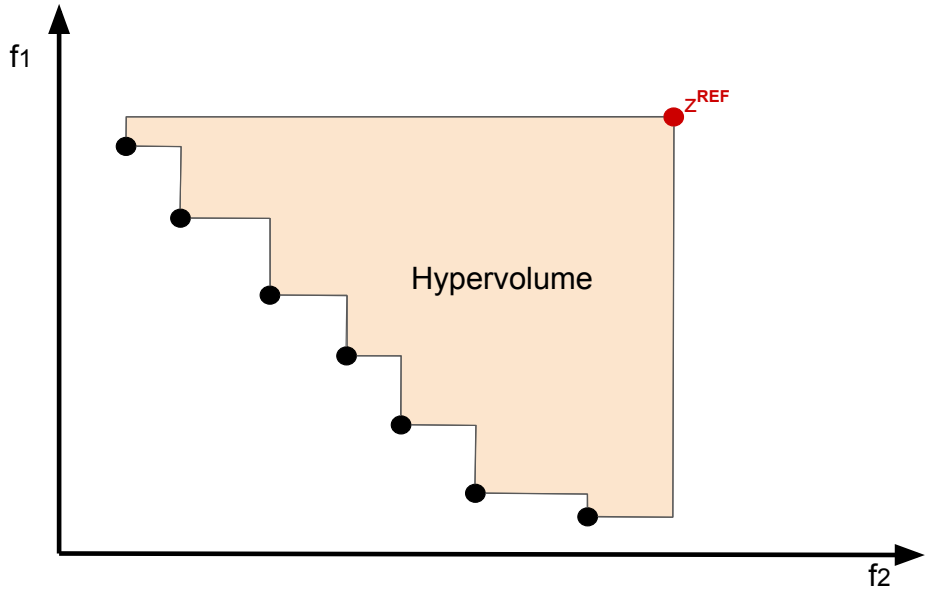

FIGURE 1.6 – Représentation du calcul de l'hypervolume pour un problème à deux objectifs.

1.5.1.2 Indicateur epsilon

Les indicateurs epsilon [121], notés habituellement I_ϵ, se basent sur la notion d'ϵ-dominance [35]. Cet indicateur existe sous deux versions : une additive et une multiplicative. Nous utilisons dans cette thèse la version additive $I_{\epsilon+}$. Cet indicateur correspond au facteur additif minimal ϵ par lequel chaque point d'une approximation A doit être translaté dans l'espace objectif pour dominer une approximation B :

$$I_{\epsilon+}(A,B) = \min_{\epsilon\in\mathbb{R}} \{\forall b \in B, \exists a \in A : a \preceq_{\epsilon+} b\} \tag{1.6}$$

où $a \preceq_{\epsilon+} b$ signifie que le vecteur objectif b ϵ-domine le vecteur a, autrement dit $\forall i \in \{1,2,\ldots,n\}, b_i \leq \epsilon + a_i$ avec $\epsilon \in \mathbb{R}^+$.

L'indicateur epsilon additif est un indicateur que l'on souhaite minimiser. Plus la valeur retournée par cet indicateur est petite, plus l'approximation du front Pareto A se rapproche de l'approximation du front B. Si la valeur est négative, cela signifie que l'approximation du front A domine l'approximation du front B. De la même manière, si l'indicateur retourne une valeur nulle alors cela signifie que les deux ensembles de solutions sont identiques. Dans nos analyses, nous utiliserons la valeur $I_{\epsilon+}(A,R)$ pour mesurer la qualité d'une approximation A par rapport au front de référence R.

1.5.2 Analyse et validation statistique des mesures de performance

Dans ce manuscrit, nous considérons des algorithmes stochastiques, c'est-à-dire que deux exécutions d'un algorithme sous les mêmes conditions peuvent donner deux résultats différents à cause des opérateurs de variations stochastiques utilisés pour générer les nouvelles solutions. Il est ainsi insuffisant d'évaluer la performance d'une métaheuristique stochastique en l'ayant exécutée une seule fois. Chacune des configurations de MOEA/D que nous étudions dans cette thèse a été exécutée dix fois indépendamment pour chaque instance de problème.

Pour comparer des mesures de performance de manière plus rigoureuse, nous allons déterminer statistiquement si un algorithme A est significativement meilleure qu'un algorithme B. Les tests statistiques non-paramétriques permettent de comparer ce genre d'échantillons de petite taille. Étant donné que nous utilisons un ensemble de graines spécifiques pour le générateur de nombres pseudo-aléatoires pour chacune des configurations mises en concurrence, nous considérons que les données sont appariées, ce qui nous permet d'utiliser un test de Wilcoxon [103] avec un seuil d'importance $\alpha = 0.05$. Ce test est un test statistique non-paramétrique qui détermine si les médianes de deux échantillons sont proches. Ce test nous permet ainsi de calculer le nombre de d'algorithmes ou de configurations qui surpassent statistiquement un algorithme particulier. On considère alors qu'un algorithme en surpasse un autre si la moyenne de ses mesures de qualité est de meilleure qualité, et si les résultats sont significativement différents selon le test de Wilcoxon.

1.5.3 Empirical attainment functions

Les fonctions d'atteinte empirique (*Empirical Attainment Functions* ou *EAF* en anglais) [65] permettent de donner une description sur la répartition des solutions obtenues par un algorithme stochastique dans l'espace objectif. Cette fonction est utilisée pour représenter graphiquement une comparaison de la distribution des solutions non-dominées entre deux algorithmes stochastiques. L'utilisation de cette méthode n'est possible que pour les problèmes à deux objectifs, car l'espace objectif est visualisé en deux dimensions.

Un exemple est disponible sur la Figure 1.7. Les zones colorées représentent les zones où la probabilité qu'un algorithme possède plus de solutions non-dominées que l'algorithme concurrent est plus importante. L'intensité de couleur change en fonction de la probabilité d'atteindre cette zone.

Nous utilisons la bibliothèque R *eaf* [65] pour comparer les approximations du front

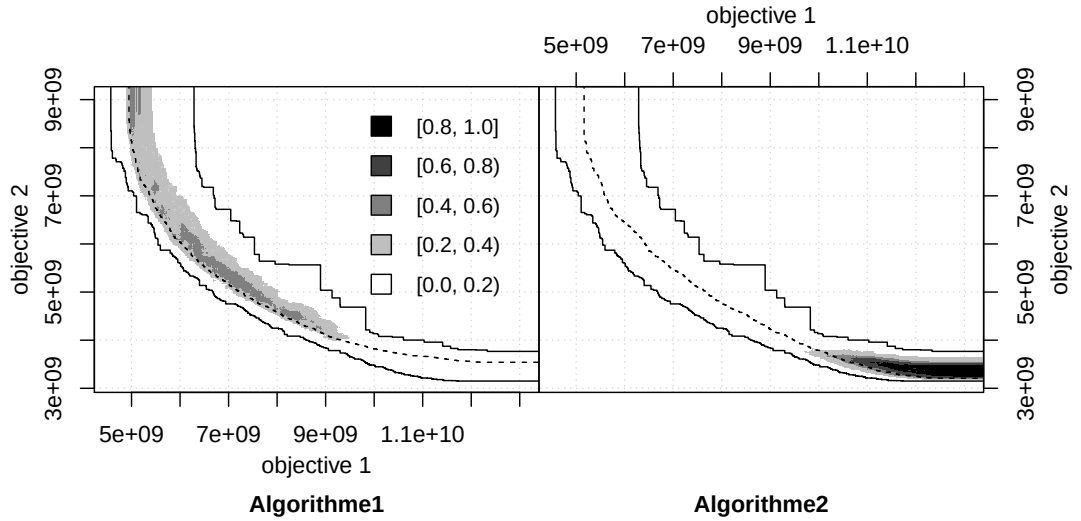

FIGURE 1.7 – Exemple d'affichage de la fonction eafdiffplot de la bibliothèque EAF

Pareto obtenues par de deux algorithmes différents.

1.5.4 Problèmes étudiés

Cette section présente les différentes classes de problèmes considérées pour l'analyse empirique des algorithmes étudiés dans cette thèse. Nous nous intéressons à deux classes de problèmes d'optimisation pseudo-booléens multi-objectif. L'intérêt de ces deux classes vient du fait qu'elles nous permettent d'étudier nos algorithmes sur des instances de problèmes avec des caractéristiques différentes qui sont détaillés ci-dessous. Nous pouvons ainsi étudier différentes tailles de problèmes, différents degrés d'interaction entre les variables et différents nombres d'objectifs à optimiser.

1.5.4.1 mnk-landscapes

La classe de problèmes mnk-landscapes [1] est une extension de la classe de problèmes d'optimisation combinatoire NP-difficile appelés nk-landscapes [46]. Les solutions sont des chaînes binaires de taille fixe. Les paysages mnk-landscapes permettent d'étudier simultanément l'influence de différentes propriétés de problèmes d'optimisation combinatoires multi-objectifs. La dimension des solutions est étudiée en modifiant la valeur du paramètre n, il s'agit du nombre de variables binaires dans l'espace de décision. Augmenter le nombre de variables dans l'espace de décision va logiquement augmenter le nombre de solutions dans l'espace de recherche (2^n) et va ainsi nous

permettre d'étudier le passage à l'échelle des algorithmes. Le paramètre k représente le degré d'interaction entre les variables, c'est-à-dire le nombre de variables qui vont influencer chaque variable d'une solution. Ce paramètre est utilisé pour gérer la rugosité du paysage. En effet, plus la valeur de k est élevée, et plus le problème est rugueux ou vallonné. Ainsi, le fait d'augmenter le degré d'épistasie k augmente alors la difficulté du problème. Enfin, le paramètre m définit le nombre d'objectifs à optimiser simultanément. Optimiser un problème mnk-landscapes revient alors à maximiser la fonction suivante avec les paramètres $m \geq 2$, $n \geq 1$ et $0 \geq k \geq n-1$:

$$f_{nk}^{j}(x) = \frac{1}{n}\sum_{i=1}^{n} f_i^j\left(x_i, x_{i_1^j}, \ldots, x_{i_{k_j}^j}\right), \forall j \in \{1, \ldots, m\} \tag{1.7}$$

où x_i représente la valeur de la variable i dans la solution x. Dans cette thèse, la même valeur d'interaction entre les variables k est utilisée pour tous les objectifs.

1.5.4.2 Le problème MUBQP

Le problème *Unconstrained Binary Quadratic Programming* nommée couramment UBQP est un problème combinatoire NP-difficile. Ce problème est introduit en 1968 par Hammer et Rudeanu [30]. Il est étudié dès les années 1970 [52] pour ses applications dans la finance [53], l'analyse économique [31] ou encore la gestion de trafic [105]. Ce problème est également étudié dans un contexte multi-objectif [62, 114, 108] où l'on souhaite optimiser simultanément m fonctions UBQP. On nommera le problème multi-objectif MUBQP.

Chaque fonction f_ℓ que l'on cherche à maximiser est définie formellement par l'équation suivante :

$$f_\ell(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}^{\ell} x_i x_j,\ \ell \in \{1, ..., m\} \tag{1.8}$$

où x est une solution avec une représentation binaire de taille n, telle que $x_i \in \{0,1\}$ représente la i^{eme} variable de la solution x, et q_{ij} est une matrice de taille $n \times n$ avec des valeurs qui peuvent être positives, négatives ou nulles. Chaque objectif ℓ possède une matrice q_{ij}^{ℓ} différente.

Cette classe de problème permet également d'étudier la dimension des solutions n et le nombre d'objectifs m avec une dépendance quadratique entre les variables.

1.6 Conclusion

Dans ce chapitre, différents aspects de l'optimisation combinatoire multi-objectif ont été abordés, en se focalisant sur les approches de résolution basées sur les algorithmes évolutionnaires multi-objectifs et la décomposition.

Nous avons tout d'abord donné les définitions liées à l'optimisation combinatoire multi-objectif permettant ainsi de fournir une meilleure compréhension du contexte dans lequel se trouve la thèse. Une fois le contexte posé, nous avons présenté une vue générale sur la résolution de problèmes d'optimisation multi-objectifs en détaillant les trois principales approches d'algorithmes évolutionnaires multi-objectifs, qui sont basées sur la dominance, les indicateurs de performances et la décomposition pour trouver une approximation du front Pareto.

Nous nous sommes ensuite focalisés sur les approches à base de décomposition, qui nous intéressent particulièrement dans cette thèse, en expliquant de façon détaillée les différents aspects de l'algorithme état-de-l'art MOEA/D. Le système de décomposition des problèmes multi-objectifs a été expliqué, tout comme le système de voisinage qui est une particularité de MOEA/D. L'algorithme étant basé sur un algorithme évolutionnaire, nous avons donc présenté la manière dont l'algorithme génère les solutions au cours du processus évolutionnaire, et comment la population est mise à jour afin de s'approcher de l'ensemble Pareto. Après avoir présenté l'algorithme, nous avons ensuite présenté le framework modulaire que nous avons développé au cours de la thèse. Cette implémentation nous a permis de développer et expérimenter facilement les composants de MOEA/D que nous avons étudiés dans ce manuscrit.

Pour terminer ce chapitre, nous avons abordé les méthodes d'analyses de performance et les problèmes considérés pour analyser les contributions présentées dans les chapitres suivants.

Dans le chapitre suivant, nous présenterons la première contribution de cette thèse, qui s'intéresse à la répartition des efforts de calculs entre les différents sous-problèmes de MOEA/D.

Chapitre 2

Répartition des efforts de calculs dans MOEA/D

Résumé

La première contribution de cette thèse s'appuie sur l'observation simple que MOEA/D, dans sa variante originale, induit une répartition équitable de l'effort de calcul entre les différents sous-problèmes définis par la décomposition. Cependant, d'autres variantes de l'algorithme ont été développées afin de différencier l'effort de calcul en fonction des besoins en optimisation de chaque sous-problème. Dans ce chapitre, nous proposons de revoir la conception de MOEA/D à la lumière de l'effort de calcul nécessaire au maintien d'une population permettant d'obtenir une bonne qualité d'approximation. À cet effet, nous proposons une nouvelle version générique de MOEA/D qui met en évidence et qui nous permet d'étudier de façon plus explicite des composants jusque-là induits de façon implicite dans les versions précédentes du framework. Notre but est de permettre une compréhension plus systématique de facteurs importants dans la conception de l'algorithme, à savoir, l'impact du nombre de sous-problèmes définis par décomposition, le nombre de sous-problèmes optimisés à chaque génération et la méthode de sélection de ces sous-problèmes. Ces travaux ont fait l'objet d'une publication à la conférence internationale EvoCOP 2020 [82].

Dans la Section 2.1 de ce chapitre, nous commençons par une discussion générale autour de la répartition de l'effort de calcul induit par MOEA/D, et de son importance dans le processus d'optimisation multi-objectif, à travers une revue des variantes dédiées de la littérature. Nous proposons ensuite dans la Section 2.2 une nouvelle conception des composants de base du framework de MOEA/D permettant une meilleure représentation

des différents éléments impactant la répartition de l'effort de calcul. Cette nouvelle conception, appelée MOEA/D-(μ, λ, sps), est étudiée de façon approfondie à travers une analyse empirique rigoureuse dans la Section 2.3.

2.1 Contexte et motivations

Dans le but d'identifier une bonne approximation du front Pareto, il est nécessaire pour tout algorithme évolutionnaire de trouver des solutions qui représentent au mieux les différentes régions de ce front. Il est donc important que le processus d'optimisation inhérent à un algorithme évolutionnaire puisse répartir l'effort de calcul de façon convenable afin d'approcher au mieux ces différentes régions. Dans cette thèse, nous considérons un budget global limité en termes du nombre d'évaluations disponibles, c'est-à-dire du nombre d'appels à la fonction permettant l'évaluation des valeurs objectif d'une solution donnée. La répartition de l'effort de calcul se matérialise ainsi par les différents mécanismes régissant les appels à la fonction d'évaluation tout au long du processus d'optimisation. Dans le contexte du framework MOEA/D, la répartition de l'effort de calcul se manifeste implicitement par la façon avec laquelle les différents sous-problèmes sont définis par la décomposition, et de façon tout aussi implicite, par la façon avec laquelle les solutions correspondant aux sous-problèmes évoluent au cours des différentes générations. Ceci est discuté de façon plus détaillée dans les paragraphes suivants.

2.1.1 La population dans MOEA/D

Dans sa conception initiale, MOEA/D induit une correspondance un-à-un entre les sous-problèmes et les solutions de la population (notée $\mathcal{P}_\mu$ dans l'Algorithme 1 du chapitre précédent). Autrement dit, chaque solution est attachée à un seul sous-problème, et inversement. Ceci est motivé par l'idée de trouver, de façon coopérative, la solution couvrant au mieux la région du front représentée par le sous-problème correspondant. En conséquence, le premier aspect pouvant impacter la distribution de l'effort de calcul dans MOEA/D est la définition même de l'ensemble des sous-problèmes obtenus par la décomposition. Comme mentionné dans le chapitre précédent, il existe plusieurs méthodes de la littérature permettant de spécifier différentes fonctions d'agrégation, ainsi que plusieurs techniques de génération des vecteurs de poids pour paramétrer ces fonctions [87, 100, 107]. Néanmoins, la plupart de ces méthodes nécessitent la spécification d'un paramètre de base qui est le nombre de sous-problèmes, ou de façon

équivalente, la taille de la population utilisée par le processus d'optimisation (noté μ dans l'Algorithme 1). Ce paramètre impact de façon implicite la répartition de l'effort de calcul du framework MOEA/D, dans la mesure où le budget global disponible est distribué en fonction de ces sous-problèmes. À budget égal, plus de sous-problèmes sont disponibles, plus l'effort de calcul dédié à chacun des sous-problèmes diminue, et inversement. Intuitivement, dans le cas où le budget global est très grand, ceci ne pose pas de problème particulier, dans la mesure où l'on peut raisonnablement penser que chaque sous-problème pourra se voir attribuer un effort de calcul suffisant pour que la solution qui lui est attachée puisse atteindre une bonne qualité. Cependant, plus le budget global diminue, plus le choix initial du nombre de sous-problèmes et le mécanisme sous-jacent de répartition de l'effort de calcul peuvent avoir un impact sur la qualité de l'approximation obtenue.

De façon plus générale, comme dans n'importe quel algorithme évolutionnaire, la taille de la population joue un rôle important [104]. D'un côté, si le nombre de solutions dans la population est trop petit, il n'est pas possible de couvrir correctement le front Pareto. Par rapport à la répartition de l'effort de calcul dans MOEA/D, ceci se traduit par le fait qu'un trop petit nombre de sous-problèmes pourraient ne pas permettre de guider la recherche vers un ensemble de solutions suffisamment représentatives du front Pareto. D'un autre côté, si la taille de la population est trop grande, le processus de recherche peut être ralenti par la perte inutile des évaluations au cours de la recherche. En effet, dans le cas de MOEA/D, un nombre trop important de sous-problèmes peut rendre beaucoup d'entre eux inutiles, car pas suffisamment différents les uns des autres. On se retrouve alors avec une population ayant des solutions trop proches les unes des autres, et donc une perte d'efficacité dans l'identification de solutions améliorantes dans les régions du front qui en auraient le plus besoin. De même, on peut noter qu'une petite taille de population pourrait permettre de se rapprocher du front Pareto de façon plus efficace, car l'effort de calcul est ainsi plus important pour chaque sous-problème. Cependant, un nombre plus important de sous-problèmes permettra certainement de mieux couvrir la totalité du front Pareto.

Bien que l'on puisse trouver différentes études traitant de l'impact de la taille de la population dans les algorithmes évolutionnaires [104, 41, 96, 38], le choix de la taille de population reste un problème difficile. À titre d'exemple, dans [41] et [96], les auteurs montrent que le réglage de ce paramètre est extrêmement compliqué et dépend du scénario d'optimisation dans lequel l'algorithme se trouve. Des stratégies adaptatives pour le choix d'une taille de population adéquate ont également été étudiées dans la littérature, par exemple avec SMS-EMOA [8], un algorithme évolutionnaire

multi-objectif basé sur un indicateur. Néanmoins, le lien entre la taille de population et la répartition de l'effort de calcul n'a été que très rarement étudié, en particulier dans le contexte de MOEA/D et plus généralement de l'optimisation multi-objectif basée sur la décomposition. Nous montrerons plus loin dans ce chapitre que des mécanismes simples vont nous permettre un choix relativement robuste du nombre de sous-problèmes à considérer dans MOEA/D.

Comme mentionné ci-dessus, outre le choix du nombre de sous-problèmes définis par la décomposition, la façon avec laquelle les solutions correspondant aux sous-problèmes évoluent au cours de la recherche est un élément important dans la répartition de l'effort de calcul de MOEA/D. En effet, chaque sous-problème est optimisé de manière itérative à chaque génération (ligne 6 de l'Algorithme 1). Lors d'une génération, on peut alors considérer que l'on génère autant de solutions qu'il y en a dans la population $\mathcal{P}$, c'est-à-dire μ solutions. En parcourant tous les sous-problèmes à chaque génération, l'effort de calcul est donc distribué équitablement entre les sous-problèmes. Il faut toutefois noter qu'un parcours itératif déterministe peut tout de même favoriser ou défavoriser certains sous-problèmes. Pour éviter tout biais, les implémentations de référence de MOEA/D parcourent en général les sous-problèmes dans un ordre aléatoire à chaque génération, ceci pour éviter de favoriser certaines régions du front Pareto, et ainsi assurer une répartition plus uniforme et homogène de l'effort de calcul [1]. Néanmoins, une répartition uniforme n'est pas toujours souhaitable si l'on suppose que l'état d'avancement de l'optimisation est différent pour chaque sous-problème considéré. Ceci est notamment lié à la difficulté relative de résolution de chaque sous-problème, qui peut amener à la conception de stratégies visant à choisir quel(s) sous-problème(s) sont à considérer à chaque génération de MOEA/D. Il s'agit là d'un aspect qui a été étudié dans de nombreux travaux de la littérature, et auxquels nous consacrons la section suivante. Plus loin dans ce chapitre, nous montrerons que le nombre de sous-problèmes sélectionnés à chaque génération, combiné au paramétrage de la taille de la population, est un élément déterminant pour une répartition efficace de l'effort de calcul de MOEA/D.

1. En effet, la solution x^i générée pendant le parcours du sous-problème i peut remplacer, en cas d'amélioration, les solutions des sous-problèmes voisins $\mathcal{B}_i$. La solution x^{i+1} générée pendant le parcours du sous-problème suivant $i+1$, qui a un voisinage $\mathcal{B}_{i+1}$ similaire au voisinage $\mathcal{B}_i$, peut être amenée à remplacer moins de solutions que la solution x^i qui aura déjà améliorée juste avant les mêmes solutions voisines. Si les sous-problèmes sont parcourus dans un ordre fixe, le front obtenu par l'algorithme peut alors être avantagé vers une extrémité et être désavantagé vers une autre extrémité du front Pareto.

2.1.2 Répartition adaptative des efforts de calcul

Divers travaux de la littérature [117, 86, 60] ont montré que différents sous-problèmes obtenus par une même méthode de décomposition, mais ayant des vecteurs de poids différents, peuvent avoir des degrés de difficultés intrinsèques différents. Bien que cela ait été le plus souvent étudié dans le contexte de problèmes d'optimisation à variables continues, il est intéressant de discuter de ces travaux, et de rappeler les mécanismes qui ont pu être mis en évidence dans le but d'obtenir une meilleure répartition de l'effort de calcul. En particulier, la différence de difficulté entre les sous-problèmes peut ralentir le processus d'optimisation évolutionnaire si un budget conséquent est inutilement consacré aux sous-problèmes les plus faciles, au détriment des sous-problèmes les plus difficiles. Zhang et al. [117] ont été les premiers à proposer l'idée de ne plus optimiser tous les sous-problèmes de manière itérative, mais plutôt de sélectionner dynamiquement un ensemble de sous-problèmes considérés comme difficiles à chaque génération. D'autres travaux attaquent le problème sous un angle différent, en proposant de modifier directement les coefficients des vecteurs de poids afin de re-diriger les sous-problèmes inutiles ou peu utiles vers des zones plus compliquées du front Pareto [86, 60]. Cette approche a été prouvée avantageuse pour éviter de perdre un effort de calcul sur les parties discontinues du front Pareto, pour lesquelles plusieurs sous-problèmes peuvent être dédiés dans la version originale de MOEA/D.

Dans cette thèse, nous nous intéresserons uniquement aux approches basées sur la sélection des sous-problèmes à optimiser, étant donné que nous sommes dans un contexte combinatoire boite-noire où la forme et la "discontinuité" du front Pareto ne soulèvent pas de problèmes particuliers. Dans les paragraphes suivants, nous présentons différentes variantes de MOEA/D issues de la littérature proposant une répartition adaptative de l'effort de calcul en se basant sur un choix dynamique des sous-problèmes à chaque génération. Toutes ces variantes ont vu le jour dans le but d'allouer l'effort de calcul aux sous-problèmes qui en ont le plus besoin à une étape donnée du processus de recherche.

2.1.2.1 MOEA/D–DRA

MOEA/D–DRA [117] est la première variante de MOEA/D qui ajoute de façon explicite un système d'allocation de l'effort de calcul en prenant en considération la difficulté de résolution des différents sous-problèmes au cours de la recherche. Cette variante est par la suite devenue un algorithme état-de-l'art pour l'allocation dynamique de l'effort de calcul dans le framework MOEA/D. Plus concrètement, au lieu de parcourir

tous les sous-problèmes au cours d'une génération (ligne 6 de l'Algorithme 1), MOEA/D–DRA parcours uniquement un sous-ensemble de sous-problèmes, considérés comme difficiles au moment de leur sélection ou ayant besoin de plus d'effort de calcul en comparaison des autres sous-problèmes.

La détection des sous-problèmes difficiles dans MOEA/D–DRA est faite en calculant une valeur d'utilité π^i pour chaque sous-problème. Cette valeur, comprise entre 0 et 1, représente l'utilité du sous-problème durant le processus de recherche. L'idée est alors de considérer que plus la valeur d'utilité d'un sous-problème est élevée, et plus celui-ci sera considéré comme important (ou prioritaire) à optimiser. En pratique, en fonction des W dernières générations, avec W un paramètre choisi initialement, un certain nombre de sous-problèmes sont sélectionnés en fonction de π^i pour entrer dans le processus d'optimisation de MOEA/D–DRA. Plus précisément, la valeur d'utilité π^i de chaque sous-problème i est calculée en fonction de l'amélioration, notée Δ_i, de la valeur obtenue par la fonction d'agrégation $g(x_i, w_i)$ au cours des W dernières générations. Si l'amélioration n'est pas suffisante, c'est-à-dire si la valeur Δ_i descend en dessous d'un seuil ϵ défini en paramètre, le sous-problème est considéré comme non-prioritaire. Sa solution x_i est considérée comme ayant mieux convergé vers le front Pareto que les autres solutions, et s'avère donc relativement peu utile à la génération actuelle. Les valeurs d'utilité sont alors mises à jour de façon continue et le processus de sélection des sous-problèmes à optimiser est ainsi répété tout au long de l'optimisation. En parallèle, seul un sous-ensemble de sous-problèmes, de taille prédéfinie initialement, est sélectionné de façon probabiliste en fonction des valeurs d'utilité π^i ainsi calculées de façon dynamique. Dans MOEA/D–DRA, $\mu/5$ sous-problèmes, dont les sous-problèmes situés aux extrémités du front Pareto, sont sélectionnés à chaque génération, via un tournoi de taille 10. Bien que le choix des paramètres concernant la façon avec laquelle les sous-problèmes sont sélectionnés ainsi que leur nombre n'aient pas fait l'objet d'une étude approfondie dans [117], on peut remarquer que le nombre de sous-problèmes sélectionnés rappelle la règle dite du "un-cinquième" (*one-fifth rule*), issue des stratégies d'évolution de type $(\mu + \lambda)$ [3].

2.1.2.2 MOEA/D–AMS

MOEA/D–AMS est une variante de MOEA/D proposée dans [17]. Cette variante fait également l'hypothèse que certains sous-problèmes ont besoin de plus d'effort de calcul que d'autres pour être optimisés. L'approche adoptée dans MOEA/D–AMS est, à cet égard, similaire à MOEA/D–DRA, avec quelques différences que nous expliquons

brièvement ci-dessous. Tout d'abord, au lieu de chercher à identifier les problèmes difficiles, c'est-à-dire les problèmes qui ont besoin de plus de ressources de calcul, MOEA/D–AMS cherche à identifier les sous-problèmes faciles. Les sous-problèmes ainsi identifiés sont alors désactivés. Un sous-problème est considéré comme facile ou résolu, s'il n'a pas été amélioré pendant les α dernières générations, avec α un paramètre prédéfini. MOEA/D–AMS désactive alors l'optimisation des problèmes identifiés comme faciles et, de ce fait, ne les parcourt plus jusqu'à leur réactivation future éventuelle. Dans MOEA/D–AMS, un sous-problème peut être réactivé s'il a été amélioré par un voisin, ou si sa solution associée fait partie des β individus avec la distance de crowding la plus élevée dans la population. La distance de crowding d'une solution représente la distance moyenne entre la solution et chacune de ses solutions voisines. Les solutions avec une distance de crowding parmi les plus élevées indiquent que les régions auxquelles elles appartiennent n'ont pas été autant explorées que les autres régions du front Pareto, ce qui implique une activation systématique de ces sous-problèmes. La désactivation et la réactivation des sous-problèmes sont réalisées en continu tout au long du processus de recherche. Ceci permet d'avantager de façon dynamique les sous-problèmes actifs et ainsi de leur affecter un effort de calcul plus conséquent.

2.1.2.3 MOEA/D–GRA et les fonctions d'utilités

MOEA/D–GRA [119] est une autre variante de MOEA/D qui se veut généraliser et étendre MOEA/D–DRA [117] et MOEA/D–AMS [17]. En effet, il y est proposé une tentative pour homogénéiser le mécanisme de sélection dynamique des sous-problèmes. L'utilité de chaque sous-problème π^i de MOEA/D–DRA est remplacé par une probabilité d'amélioration p^i. Les sous-problèmes ne sont plus choisis par un tournoi basé sur leur valeur d'utilité, mais directement en fonction de cette probabilité de sélection. Ainsi, la notion de *fonction d'utilité* y est introduite de façon plus explicite. Une fonction d'utilité, selon la terminologie utilisée dans MOEA/D–GRA, est une fonction utilisée pour calculer l'importance de chaque sous-problème. Par exemple pour MOEA/D–DRA, la fonction d'utilité appelée *amélioration relative* représente le calcul de la différence d'amélioration Δ_i, comme discuté auparavant. La probabilité d'amélioration p^i, inspiré par les mécanismes de MOEA/D–DRA, est alors calculé dans MOEA/D–GRA de la façon suivante :

$$p^i = \frac{\Delta^i + \varepsilon}{\max_{j=1,\cdots,N} \left\{\Delta^j\right\} + \varepsilon} \tag{2.1}$$

où ε est une très petite valeur pour éviter les divisions par zéro. Le principe de MOEA/D–GRA reste le même que pour MOEA/D–DRA ou MOEA/D–AMS : si un sous-problème a été significativement amélioré pendant les dernières générations, sa probabilité d'être sélectionné doit être plus importante, car cela signifie qu'il a plus de potentiel que d'autres sous-problèmes et mérite donc plus de ressources de calcul [2].

MOEA/D–GRA permet de considérer facilement de nouvelles fonctions d'utilité. Ces fonctions permettent d'extraire diverses informations concernant les sous-problèmes, notamment la difficulté à être optimisé à un moment précis du processus de recherche. En plus de la fonction d'amélioration relative issue de MOEA/D–DRA, six nouvelles fonctions sont proposées. La première est basée sur la densité des solutions dans l'espace objectif. Le but est de sélectionner les sous-problèmes où les régions sont peu denses en solutions, afin de mieux couvrir le front Pareto. La deuxième fonction est une variante de la première. Au lieu de se baser sur la distribution des solutions dans l'espace objectif, cette fonction se base sur la distribution dans l'espace de décision. Avec cette fonction, le but est ainsi d'améliorer la diversité de la population. Les quatre dernières fonctions sont des fonctions hybrides, formées en utilisant ces deux premières fonctions. Plus récemment, Lavinas et al. [54] ont également proposé deux nouvelles fonctions basées sur la diversité dans l'espace de décision. La première fonction a pour but de sélectionner les sous-problèmes pour donner plus de ressources de calcul aux solutions qui diffèrent le plus de leurs parents, et ainsi obliger le processus de recherche à visiter des zones moins explorées. La deuxième fonction est l'inverse de la première, c'est-à-dire que les sous-problèmes sont sélectionnés de telle sorte qu'ils soient "proches" de leurs solutions, afin d'exploiter plus intensément l'espace de recherche.

Nous concluons cette revue des méthodes portant sur la sélection dynamique de sous-problèmes pour l'allocation de l'effort de calcul en notant que la plupart d'entre elles ont été proposées dans le cadre de problèmes d'optimisation à variables continues. Cependant, leur étude n'a pas été approfondie pour des problèmes combinatoires, à variables discrètes. La suite de ce chapitre présente notre contribution dans l'amélioration et l'étude du framework de MOEA/D dans le contexte de l'optimisation combinatoire multi-objectif.

2. Notons, à ce niveau de notre présentation, que cette hypothèse peut toutefois être remise en question, notamment lorsque le processus d'optimisation s'avère être bloqué dans certaines régions sous-optimales. Dans ce chapitre, nous nous contentons de décrire les mécanismes de la littérature permettant une meilleure distribution de l'effort de calcul.

2.2 Étude de l'allocation des efforts de calcul dans MOEA/D

Dans la suite de ce chapitre, nous présentons la première contribution de cette thèse, dont le but est de permettre de mieux expliciter les différents mécanismes et paramètres pouvant impacter la conception de techniques efficaces pour la répartition de l'effort de calcul au sein du framework MOEA/D.

2.2.1 MOEA/D-(μ, λ, *sps*) : Une reformulation simple pour l'allocation des ressources dans MOEA/D

Le système d'allocation des ressources mis en avant par MOEA/D–DRA, MOEA/D–AMS ou MOEA/D–GRA est un élément important des algorithmes évolutionnaires multi-objectifs par décomposition. Cependant, ces travaux se sont focalisés sur la détection des sous-problèmes difficiles (ou de façon équivalente, faciles) dans l'idée d'allouer plus d'effort de calcul (c'est-à-dire plus d'appels à la fonction d'évaluation) à ces sous-problèmes. Dans notre contribution [82], nous proposons d'étudier le système d'allocation des ressources suivant trois éléments distincts, mais complémentaires, dont la spécification conjointe peut impacter la qualité de la recherche. Plus précisément, la fonction de sélection des sous-problèmes (fonction d'utilité), sur laquelle s'est focalisé l'ensemble des travaux existants, n'est pas le seul élément important. Comme nous allons le montrer par la suite, le choix de la taille de la population, ainsi que le choix du nombre de sous-problèmes parcourus à chaque génération, sont également deux paramètres fondamentaux pour gérer correctement l'effort de calcul dans MOEA/D. Nous proposons donc de mener une étude rigoureuse permettant de mieux comprendre l'effet combiné de la taille de population, du nombre de sous-problèmes sélectionnés, et de la stratégie de sélection des sous-problèmes, à travers une variante simple et générique de MOEA/D, que l'on note MOEA/D-(μ, λ, *sps*) dans la suite.

Le framework de MOEA/D-(μ, λ, *sps*) que nous proposons est décrit dans le template de l'Algorithme 2. Comme nous pouvons le remarquer, cet algorithme ne diffère que très peu de la version de base de MOEA/D, tel que décrit dans le chapitre précédant à travers l'Algorithme 1. La différence principale se situe au niveau de la fonction de sélection des sous-problèmes $\mathcal{W}_\lambda$ à optimiser pendant une génération (ligne 6), notée *sps*, ainsi que dans la spécification explicite des deux paramètres μ et λ, que nous considérons donc comme des paramètres fondamentaux. En effet, dans MOEA/D-(μ, λ, *sps*), la taille de la population μ et le nombre de sous-problèmes parcourus à chaque génération λ sont clairement dissociés de la stratégie de sélection. Il est à noter que ceci n'était pas le cas

dans les variantes existantes telles que MOEA/D–DRA, où le nombre de sous-problèmes à sélectionner n'est pas étudié en tant que tel et de façon indépendante de la stratégie de sélection. Notons aussi que la notation (μ, λ, *sps*) est inspirée de la notation standard des algorithmes évolutionnaire ($\mu + \lambda$), où μ représente la taille de la population et λ représente le nombre de nouveaux individus générés avant l'étape de remplacement permettant la création de la nouvelle population à chaque génération. Enfin, notons que le composant *sps* dans MOEA/D-(μ, λ, *sps*) est clairement inspiré des variantes existantes de MOEA/D tel que discuté dans la section précédente. Ce composant permet de spécifier la stratégie à utiliser à chaque génération de MOEA/D afin de choisir les λ sous-problèmes (ou de façon équivalente, les λ solutions) qui seront parcourus par l'algorithme en suivant la procédure standard de variation et de sélection de MOEA/D. Ce composant est donc similaire à la fonction d'utilité de MOEA/D–GRA.

2.2.2 Discussion et résumé des variantes considérées

De façon fondamentale, la décomposition du système d'allocation de l'effort de calcul dans MOEA/D-(μ, λ, *sps*) en trois éléments distincts (c'est-à-dire à travers les paramètres μ, λ , *sps*), permet de souligner l'interdépendance de ces trois composants en nous permettant d'étudier de manière plus précise l'impact conjoint de différentes combinaisons possibles.

En particulier, la taille de la population, ou le nombre de sous-problèmes, a maintenant un rôle différent de celui de la version standard de MOEA/D. Les vecteurs de poids peuvent dorénavant être vus comme jouant le rôle d'une structure de donnée globale organisant les solutions de la population. Cette structure globale est en interaction explicite avec la composante de sélection des sous-problèmes lors de la distribution de l'effort de calcul.

De même, le nombre λ de sous-problèmes sélectionnés et parcourus à chaque génération permet maintenant une configuration plus fine du nombre de solutions générées lors d'une génération. De façon générale, il s'agit ici d'un aspect important pour tout algorithme évolutionnaire de type ($\mu + \lambda$), où il est bien connu que le nombre de nouvelles solutions créées peut avoir un impact important sur le degré d'exploration et d'exploitation dans le processus d'optimisation évolutionnaire. Notons par ailleurs que les λ solutions enfants ne sont pas simplement générées à partir des solutions associées aux sous-problèmes sélectionnés. En effet, la sélection des solutions parentes se fait directement dans le voisinage local, qui peut comprendre des solutions associées à la fois à des sous-problèmes sélectionnés ou non sélectionnés par la stratégie *sps*.

Algorithme 2 : MOEA/D-(μ, λ, *sps*)

Input : $\mathcal{W} := \left\{w^1, \ldots, w^\mu\right\}$: ensemble de vecteurs de poids ; $g(\cdot \mid \omega)$: fonction d'agrégation ; T : taille du voisinage ; *sps* : stratégie de sélection des sous-problèmes, λ : nombre de sous-problèmes sélectionnés

$\mathcal{EP} \leftarrow \varnothing$: (Facultatif) archive externe ;

$\mathcal{P} \leftarrow \left\{x^1, \ldots, x^\mu\right\}$: générer et évaluer la population initiale de taille μ;

$z^\star \leftarrow$ initialiser le point de référence avec les solutions de la population initiale $\mathcal{P}$;

tant que *le critère d'arrêt n'est pas atteint* **faire**

/* sélection des λ sous-problèmes à parcourir parmi les μ de disponible */

$\mathcal{W}_\lambda \leftarrow$ sélection_sous-problèmes(λ, *sps*);

/* Parcours du sous-problème i */

pour $w^i \in \mathcal{W}_\lambda$ **faire**

/* Récupération du voisinage du sous-problème i avec les T plus proches sous-problèmes */

$\mathcal{B}_i \leftarrow$ voisinage(T, i);

/* Sélection des parents dans le voisinage $\mathcal{B}_i$ */

$\mathcal{X} \leftarrow$ parents($\mathcal{B}_i$);

/* Utilisation d'opérateurs de variation pour générer une solution enfant à partir de $\mathcal{X}$ */

$x' \leftarrow$ reproduction($\mathcal{X}$);

/* Evaluation de la solution enfant x' */

$F(x') \leftarrow$ évaluation(x') ;

/* Mise à jour du point de référence avec $F(x')$ */

$z^\star \leftarrow$ mise_à_jour_z($z^\star$, $F(x')$) ;

/* mise à jour de l'archive externe des solutions non-dominées avec la nouvelle solution x' */

$\mathcal{EP} \leftarrow$ mise_à_jour_archive($\mathcal{EP}$, x') ;

/* processus de remplacement de la solution x' dans le voisinage $\mathcal{B}_i$ */

pour $j \in \mathcal{B}_i$ **faire**

si $g(F(x')|w^j)$ ***est meilleur que*** $g(F(x^j)|w^j)$ **alors**

$\mathcal{P}_j \leftarrow x'$;

Enfin, la formulation du framework MOEA/D-(μ, λ, *sps*) permet de modéliser différentes variantes de MOEA/D, qu'elles soient munies d'un système d'allocation dynamique de l'effort de calcul, ou non. Par exemple, pour le framework original MOEA/D,

Tableau 2.1 – Différentes instanciations de MOEA/D-($\mu, \lambda, \mathsf{sps}$).

algorithme	taille de pop.	# sous-prob.	stratégie de sélection	ref.
MOEA/D	μ	μ	$\mathsf{sps}_{\text{All}}$	[87]
MOEA/D–DRA	μ	$\mu/5$	$\mathsf{sps}_{\text{Dra}}$	[117]
MOEA/D–RND	μ	$\lambda \leqslant \mu$	$\mathsf{sps}_{\text{Rnd}}$	[82]

le nombre de sous-problèmes sélectionnés peut être écrit avec $\lambda = \mu$ et une stratégie de sélection, notée $\mathsf{sps}_{\text{All}}$, qui retourne simplement tous les sous-problèmes disponibles. De la même manière, en prenant une valeur de λ égale à $\mu/5$, et une stratégie de sélection *sps* conforme, nous pouvons facilement reproduire l'algorithme MOEA/D–DRA tel que décrit originellement dans [117]. Nous utiliserons alors $\mathsf{sps}_{\text{Dra}}$ la stratégie de sélection permettant de reproduire l'implémentation de MOEA/D–DRA avec notre proposition.

La suite de ce chapitre est dédiée à l'étude de l'effet combiné des différents paramètres de MOEA/D-(μ, λ, sps). En particulier, nous considérons une variante notée MOEA/D–RND, dans laquelle nous employons une stratégie de sélection basique, notée $\mathsf{sps}_{\text{Rnd}}$, retournant simplement λ sous-problèmes sélectionnés aléatoirement parmi les μ disponibles. De plus, nous considérons bien sûr les variantes de référence MOEA/D et MOEA/D–DRA, tel que résumé dans le Tableau 2.1.

2.3 Analyse expérimentale de MOEA/D-(μ, λ, sps)

2.3.1 Protocole expérimental

2.3.1.1 Instances étudiées

Nous considérons dans ce chapitre un panel relativement large d'instances du problème mnk-landscapes (voir la Section 1.5.4 pour une définition formelle détaillée). Pour nos expérimentations, nous fixons la taille des instances à $n = 100$, ce qui représente des problèmes avec un espace de décision raisonnablement grand. Nous utilisons ensuite des problèmes dont le nombre d'objectifs varie de 2 à 5 objectifs : $m \in \{2, 3, 4, 5\}$. Notons ici que des problèmes avec un nombre d'objectifs plus élevé ont en général des fronts Pareto de cardinalité exponentiellement plus grande. Nous faisons également varier le degré d'épistasie k pour couvrir des instances avec un degré de difficulté variable. Plus précisément, nous considérons des instances telles que $k \in \{0, 1, 2, 4\}$. La plus petite valeur de $k = 0$ représente des problèmes linéaires, alors que des valeurs plus grandes de k représentent des problèmes quadratiques ($k = 1$), cubiques ($k = 2$), et fortement rugueux ($k = 3, 4$). Il est en effet à noter qu'augmenter le degré d'épistasie k a pour effet

connu d'augmenter le nombre d'optima locaux, ce qui rend les instances générées de plus en plus difficiles.

2.3.1.2 Algorithmes et paramètres

Nous considérons pour notre analyse trois implémentions de MOEA/D-(μ, λ, *sps*), comme décrit dans le Tableau 2.1. L'algorithme MOEA/D, aussi noté MOEA/D-(μ, μ, sps_{ALL}), est l'algorithme de base qui nous permettra de déterminer les éventuels apports de MOEA/D-(μ, λ, *sps*). MOEA/D-(μ, μ, sps_{ALL}) dispose d'un seul paramètre laissé libre, il s'agit de la taille de la population μ. L'algorithme MOEA/D–DRA fait figure d'état-de-l'art pour les algorithmes à base de décomposition avec allocation dynamique de l'effort de calcul. La stratégie de sélection des sous-problèmes est implémentée comme dans l'article [117] et peut être vue comme la fonction d'utilité *d'amélioration relative* dans les articles [119, 55]. Le nombre de sous-problèmes sélectionnés dans l'article original est de $\lambda = \mu/5$ (en incluant les sous-problèmes situés aux extrémités). L'algorithme MOEA/D–RND est une variante basique de comparaison, ayant une stratégie extrêmement simple de sélection des sous-problèmes. La stratégie de sélection sps_{RND} est implémentée de la même manière que la stratégie sps_{DRA}, dans le sens où les vecteurs de poids situés aux extrémités seront systématiquement ajoutés, ceci afin de comparer les deux stratégies le plus équitablement possible.

La taille de population μ est paramétrable pour les trois implémentations étudiées. Nous avons choisi de considérer un ensemble de valeurs pour étudier des configurations allant de très petites populations à de relativement grandes populations, $\mu \in \{10, 50, 100, 500\}$. Le paramètre λ, qui représente le nombre de sous-problèmes sélectionnés à chaque génération, est paramétrable pour seulement deux des trois algorithmes implémentés : MOEA/D–RND et MOEA/D–DRA. La valeur de λ doit être strictement inférieure à la taille de la population. Si $\lambda = \mu$, tous les sous-problèmes seront parcourus ce qui rend la stratégie de sélection des sous-problèmes inutile et nous renvoie à la version originale de MOEA/D. Les valeurs testées sont les suivantes, $\lambda \in \{1, 2, 5, 10, 25, 50, 100, 150, 200, 300, 400, 450, 500\}$.

Pour les paramètres communs aux trois implémentations, nous avons choisi d'utiliser le réglage le plus standard de la littérature. Les vecteurs de poids sont générés en suivant la méthodologie SOBOL [115]. La taille du voisinage T est fixée à 20% de la taille de la population, c'est-à-dire $T = 0.2\ \mu$. Afin de générer de nouvelles solutions, nous utilisons comme parents la solution associée au sous-problème actuellement parcouru et une solution choisie aléatoirement dans le voisinage du sous-problème courant. Avec

ces deux solutions parentes, une solution enfant est générée grâce à un croisement standard à 2 points, suivi d'un opérateur de mutation standard qui modifie chaque variable booléenne avec un taux de $1/n$. Les paramètres supplémentaires de MOEA/D–DRA, comme δ ou nr, sont configurés en suivant les recommandations de l'article original [117].

2.3.1.3 Évaluation des performances

En considérant chaque combinaison de paramètres à étudier pour chaque instance de mnk-landscapes, il existe plus de 2 000 configurations différentes. Pour gérer correctement nos expérimentations en un temps raisonnable, nous exécutons chaque configuration dix fois pour un total de 20 000 exécutions. Pour visualiser correctement la tendance de chaque configuration en fonction du nombre d'évaluations, nous considérons plusieurs conditions d'arrêt pour un budget maximal de 10^7 évaluations $(10^0, 10^1, ..., 10^7)$.

L'indicateur de qualité utilisé pour étudier la qualité des solutions obtenues est celui de la *déviation relative à l'hypervolume* tel que décrit dans la Section 1.5.1.1 du Chapitre 1. L'indicateur de qualité est utilisé sur l'archive externe qui stocke toutes les solutions non-dominées trouvées au fur et à mesure du processus de recherche. L'utilisation de l'archive externe est très importante car cela nous permet de comparer des configurations différentes, même si les algorithmes étudiés utilisent des tailles de population différentes.

2.3.2 Étude de la taille de population dans MOEA/D

Le premier aspect que nous proposons d'étudier est l'effet de la taille de la population μ dans la version standard de MOEA/D, c'est-à-dire sans les nouveaux paramètres λ et sps. Comme nous l'avons expliqué précédemment, le paramètre μ de MOEA/D joue à la fois un rôle sur le nombre de solutions maintenues dans la population, mais aussi sur le nombre de solutions générées à chaque génération. Comme pour n'importe quel algorithme évolutionnaire, il est intéressant ici d'étudier son comportement par rapport aux caractéristiques des différentes instances de problèmes considérées. Cette analyse nous permettra de déterminer dans les sections suivantes si l'impact de μ est toujours le même lorsque les composants de sélections des sous-problèmes seront introduits.

Dans la Figure 2.1, nous affichons les approximations obtenues lors d'une exécution de MOEA/D-(μ, μ, $\mathrm{sps}_{\mathrm{ALL}}$) pour deux budgets différents (10^4 et 10^6 évaluations). Les fronts visualisés montrent toutes les solutions non-dominées stockées dans l'archive externe que l'algorithme a pu trouver depuis le début du processus de recherche. Cette

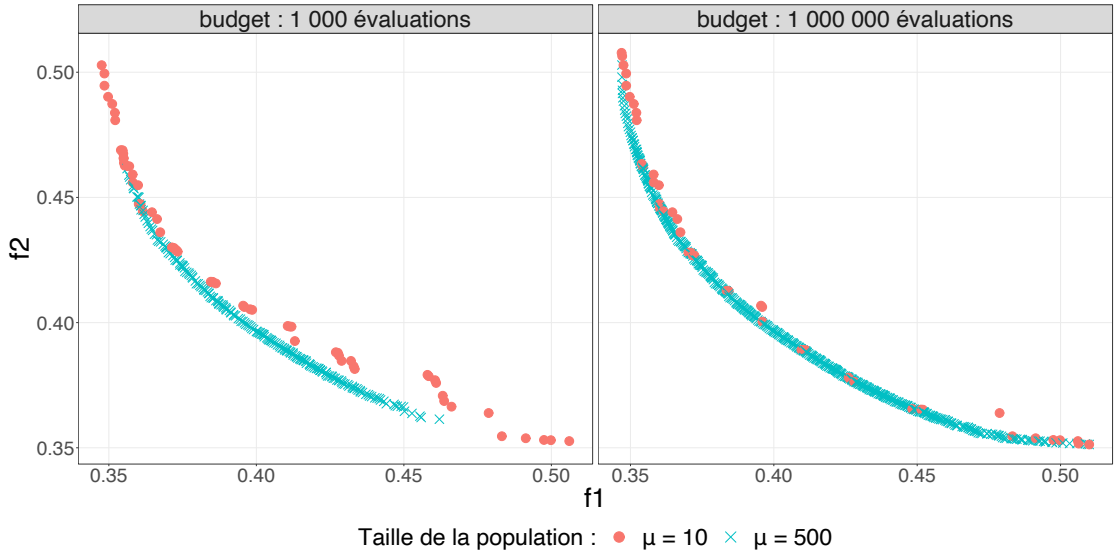

FIGURE 2.1 – Illustration du front Pareto de MOEA/D sur une instance linéaire ($k = 0$) de mnk-landscapes avec 2 objectifs ($m = 2$). Deux tailles de populations ainsi que deux budgets différents sont considérées.

figure nous permet de comparer les solutions de MOEA/D-(μ, μ, sps_{ALL}) avec une taille de population que l'on peut considérer comme petite, c'est-à dire $\mu = 10$, et une grande population, c'est-à-dire $\mu = 500$. Dans la Figure 2.2, nous mettons en évidence les profils de convergences de MOEA/D pour différentes tailles de population. Le profil de convergence correspond ici à l'évolution de la déviation relative d'hypervolume en fonction du nombre d'évaluations effectuées par l'algorithme.

Avec un budget fixe, on peut considérer que choisir une petite population permet au processus de recherche de focaliser ses efforts de calcul sur beaucoup moins de sous-problèmes qu'avec une grande population. De ce fait, on peut alors dire qu'une petite population approchera plus rapidement le front Pareto qu'une grande population, qui devra quant à elle partager ses efforts de calcul vers plus de sous-problèmes. D'un autre côté, si on considère un grand nombre de solutions dans la population, on aura alors plus de solutions différentes et donc plus de diversité. On peut donc dire qu'avec une grande population, l'approximation obtenue est beaucoup plus étendue et détaillée que si l'on a peu de solutions dans notre population. Ces affirmations sont confirmées par nos observations dans les Figures 2.1 et 2.2.

En effet, si on compare les solutions obtenues par MOEA/D-(μ, μ, sps_{ALL}) avec un faible budget (10^4 évaluations) sur la Figure 2.1, on peut voir que la plus petite population ($\mu = 10$) couvre plus vite l'ensemble du front Pareto que la plus grande

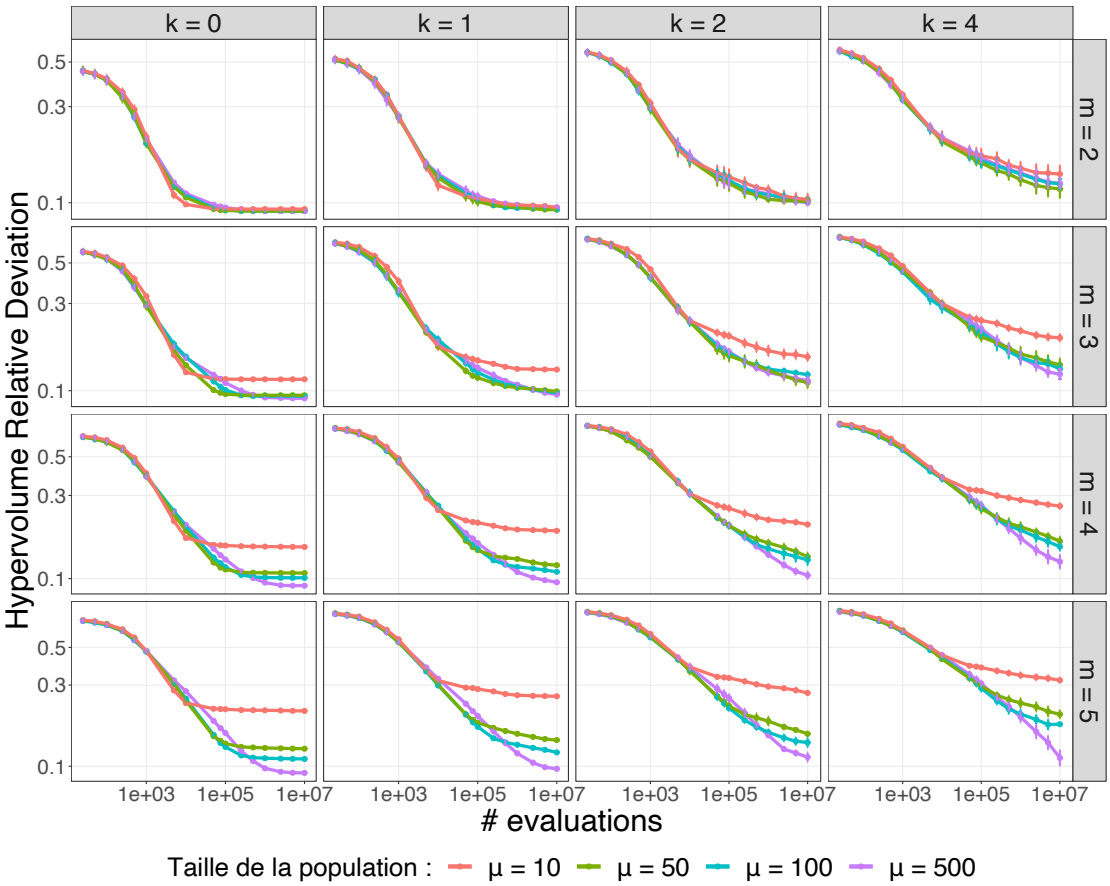

FIGURE 2.2 – Profils de convergence de MOEA/D en fonction du nombre de solutions dans sa population. Les échelles des abscisses et des ordonnées sont en log().

population ($\mu = 500$). Même si la plus grande population semble être plus avancée vers le centre du front Pareto, sa présence est quasi inexistante sur les extrémités. Dans cet exemple, on peut noter que pour un budget de 10^4 évaluations, le nombre de solutions non-dominées trouvées avec une population de taille $\mu = 10$ est de 47, alors qu'il est de 210 avec la population de $\mu = 500$ individus. Quand on augmente le budget à 10^6 évaluations, la population ayant le plus grand nombre d'individus a rattrapé son retard. En effet, une population de $\mu = 500$ individus couvre maintenant la quasi-entièreté du front Pareto, avec 507 solutions non-dominées trouvées. L'algorithme avec une population de $\mu = 10$ individus s'est légèrement rapproché sans toutefois couvrir plus de terrain, avec ses 47 solutions non-dominées. Quand on compare les tailles de population sur la Figure 2.2 avec des budgets différents allant de quelques centaines d'évaluations à plusieurs centaines de milliers d'évaluations, on peut voir que

des petites populations peuvent atteindre de meilleures solutions que des populations bien plus grandes, tout particulièrement pour les problèmes avec des paysages assez lisses, c'est-à-dire lorsque $k \leq 1$. Lorsque le budget est plus élevé, les meilleurs résultats sont obtenus par les plus grandes populations. En effet, bien qu'elles soient surpassées avec un budget peu élevé, les grandes populations ne sont jamais surpassées sous un budget suffisant. On peut également remarquer que pour les instances les plus simples, toutes les tailles de population permettent d'obtenir des performances équivalentes lorsque le budget est important. À l'inverse, pour les instances plus compliquées, seules les grandes populations permettent d'atteindre les meilleurs résultats.

On peut noter que la définition de "petite" et de "grande" population est toute relative, et dépend de l'instance de mnk-landscapes. Ainsi, une population de $\mu = 10$ individus semble se rapprocher plus rapidement du front dans les premières évaluations qu'une population de $\mu = 500$ individus pour les problèmes linéaires ($k = 0$). Néanmoins, quand on augmente le degré d'épistasie k, on peut voir qu'il existe des tailles de population qui semblent beaucoup trop petites pour obtenir de bons résultats. Par exemple pour les problèmes cubiques ($k = 2$), une population avec $\mu = 10$ individus semble insuffisante, alors qu'une population avec $\mu = 50$ ou $\mu = 100$ individus se rapprochent plus rapidement du front Pareto que la population de $\mu = 500$ individus lors des premières évaluations.

Pour résumer cette première analyse, on peut maintenant dire que le paramétrage optimal de la taille de la population de MOEA/D dépend à la fois du budget disponible et des caractéristiques du problème à optimiser. Pour de petits budgets, les petites populations sont à privilégier car elles permettent de s'approcher rapidement du front Pareto, même si elles ne le couvrent pas complètement. Cependant, quand le budget disponible est plus important et que la difficulté du problème augmente au niveau de la rugosité, les populations plus grandes obtiennent de meilleurs résultats. Grâce à ce panel d'instances, on peut aussi remarquer que l'impact du nombre de solutions dans la population augmente avec le nombre d'objectifs (le paramètre m de mnk-landscapes). Avec ces observations, il nous semble donc possible que le comportement "anytime" de MOEA/D puisse être amélioré si un choix dynamique de la taille de population était envisagé. Cependant, l'utilisation de stratégies de sélection des sous-problèmes semble aussi constituer une bonne approche. L'hypothèse qui peut être formulé ici est que réduire le nombre de sous-problèmes parcouru permettra de s'approcher plus rapidement du front Pareto, comme le fait une population avec moins d'individus dans MOEA/D. Il nous reste donc à déterminer la validité de cette hypothèse en étudiant l'impact de la stratégie de sélection des sous-problèmes *sps* et l'impact du nombre de

sous-problèmes sélectionnés λ. Mais avant cela, nous étudions dans la section suivante l'impact des sous-problèmes situés aux extrémités, qui sont ajoutés systématiquement dans la stratégie de sélection de MOEA/D–DRA.

2.3.3 Étude de l'impact des sous-problèmes situés aux extrémités

Dans MOEA/D–DRA, les auteurs incluent systématiquement les sous-problèmes situés aux extrémités aux sous-problèmes sélectionnés à chaque génération. Pour un problème à m objectifs, il existe m sous-problèmes situés aux extrémités, qui possèdent dans leur vecteur de poids un poids égale à 1 et $m-1$ poids égaux à 0. Par exemple pour un problème à 4 objectifs, les vecteurs de poids de ces sous-problèmes seront : $(1,0,0,0)$, $(0,1,0,0)$, $(0,0,1,0)$, et $(0,0,0,1)$. Le choix de sélectionner ces sous-problèmes n'est pas commenté par les auteurs, mais il pourrait s'expliquer par le fait que la fonction de sélection des sous-problèmes ne retourne pas assez souvent ces sous-problèmes, ce qui ne permettrait pas de couvrir l'entièreté du front Pareto, notamment sur les extrémités. Afin de déterminer si l'ajout de ces sous-problèmes situés aux extrémités est vraiment utile quand on parcourt moins de μ sous-problèmes par génération, notamment pour notre stratégie de base sps_{RND}, nous étudions l'impact de ces sous-problèmes sur MOEA/D-(μ, λ, sps_{RND}). Contrairement à la stratégie sps_{DRA}, la stratégie sps_{RND} donne autant de chance aux sous-problèmes situés aux extrémités qu'aux autres sous-problèmes d'être choisis. L'impact des sous-problèmes aux extrémités devrait alors être plus faible que pour la stratégie sps_{DRA}, qui peut ne pas choisir ces sous-problèmes pendant de nombreuses générations si d'autres sous-problèmes sont considérés comme plus difficiles.

La Figure 2.3 compare les profils de convergence de deux configurations de MOEA/D-($\mu = 500$, $\lambda = 1$, sps_{RND}). Les deux configurations sélectionnent $\lambda+m$ sous-problèmes avec la stratégie sps_{RND} pour comparer les deux configurations dans les mêmes conditions. Dans la configuration "avec extrémités", 1 sous-problème est sélectionné aléatoirement en plus des m sous-problèmes situés aux extrémités. Dans la deuxième configuration appelée "sans extrémités", $\lambda + m$ sous-problèmes sont sélectionnés aléatoirement. La Figure 2.3 montre très clairement que la sélection des sous-problèmes situés aux extrémités en plus des λ sous-problèmes choisis par la stratégie *sps* permet d'améliorer le profil de convergence de l'algorithme. La différence est nettement plus visible pour les instances les plus simples, que ça soit avec de faibles coefficients d'épistasie ($k \leq 1$) ou avec peu d'objectifs ($m = 2$).

Pour essayer d'expliquer ce phénomène, nous montrons dans la Figure 2.4 les fonc-

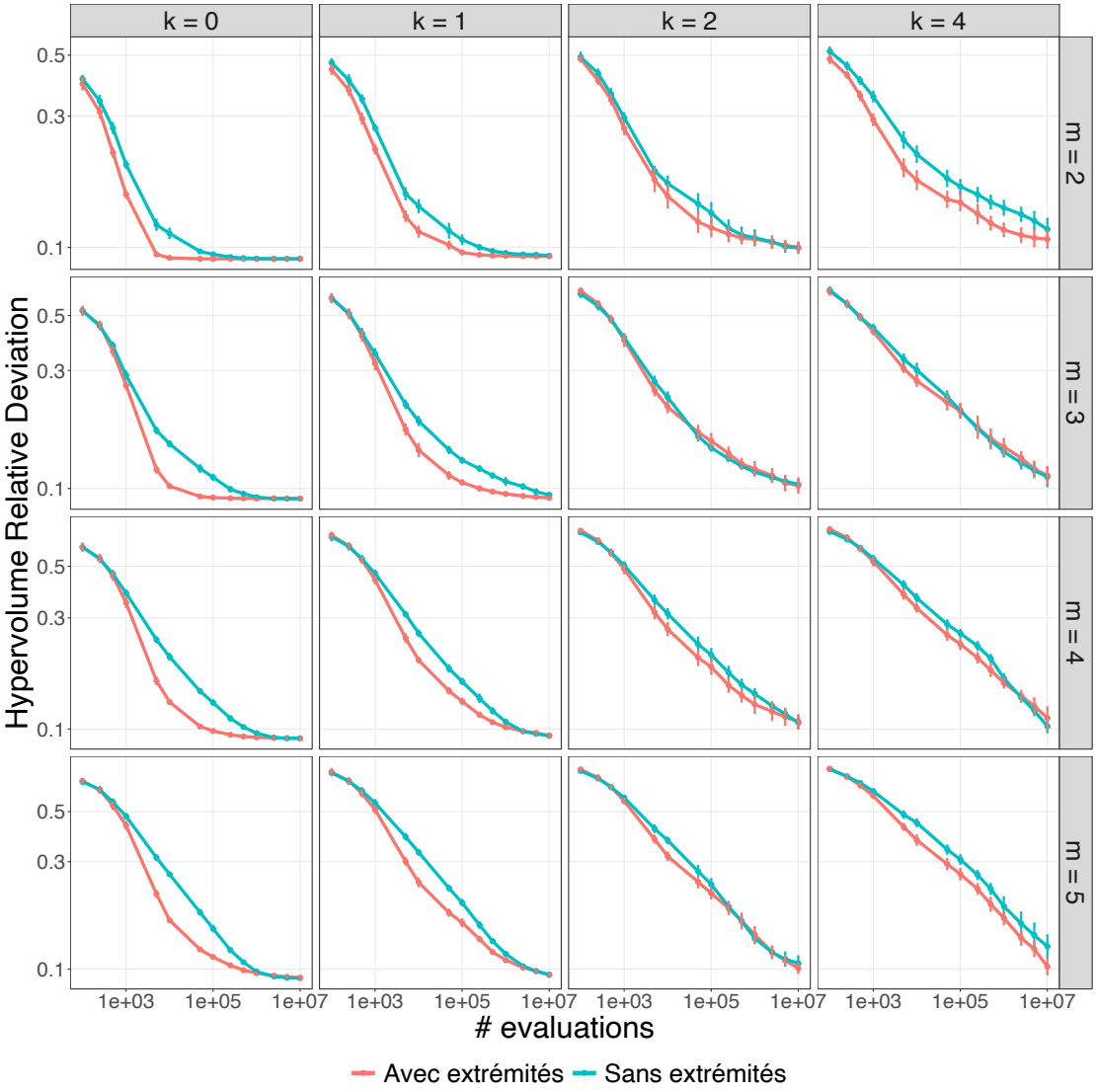

FIGURE 2.3 – Profils de convergence de MOEA/D-(μ, λ, sps_{RND}) avec $\mu = 500$. Chacune des configurations correspond à une version avec inclusion ou non des sous-problèmes situés aux extrémités. Pour la version "avec les extrémités", on sélectionne 1 sous-problème aléatoirement et on y ajoute les m extrémités. Pour la version "sans les extrémités", on sélectionne $m+1$ sous-problèmes aléatoirement.

tions d'atteinte empirique (EAF) [65] qui nous permettent de comparer les solutions obtenues par deux algorithmes sur plusieurs exécutions, ceci afin d'identifier les zones du front où un algorithme est plus performant que l'autre. Nous considérons ici une instance de mnk-landscapes avec $m = 2$ et $k = 0$, pour un budget de 10^4 évaluations sur

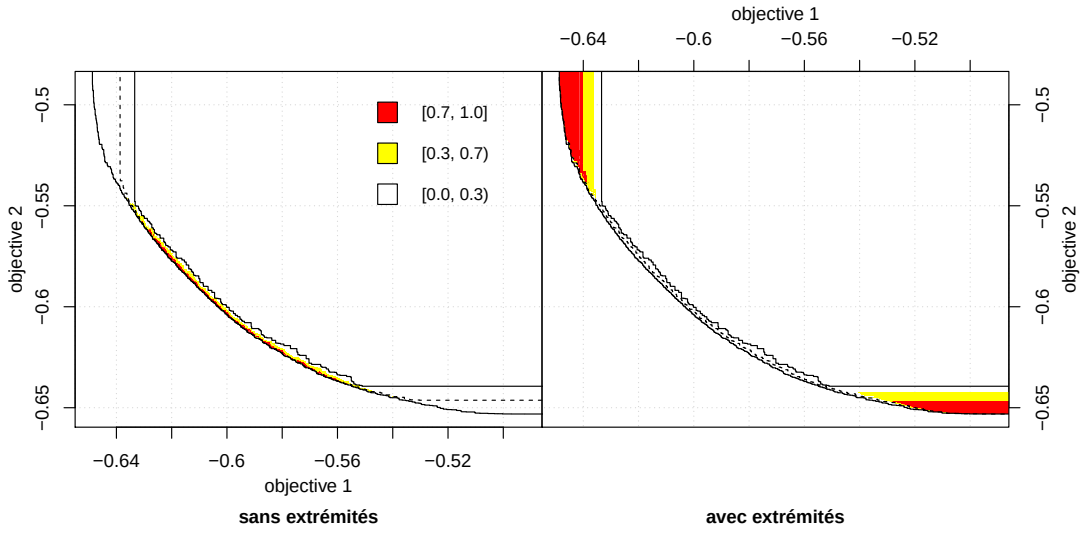

FIGURE 2.4 – EAF de MOEA/D-(μ, λ, sps_{RND}) avec $\mu = 500$ sur une instance de mnk-landscapes avec $m = 2$ et $k = 0$. Chacune des configurations correspondent à une version avec inclusion ou non des sous-problèmes situés aux extrémités. Pour la version "avec les extrémités", on sélectionne 1 sous-problème aléatoirement et on y ajoute les m extrémités. Pour la version "sans les extrémités", on sélectionne $m + 1$ sous-problèmes aléatoirement.

10 exécutions indépendantes de chaque configuration. Sur cette figure, on peut voir que la configuration qui sélectionne les extrémités domine la configuration concurrente au niveau des extrémités du front Pareto. Si les sous-problèmes situés aux extrémités ne sont pas sélectionnés à chaque génération, le processus de recherche n'arrivera pas à trouver les solutions correspondantes à l'aide des autres sous-problèmes sélectionnés aléatoirement. Le fait de ne pas trouver les solutions aux extrémités du front Pareto est un facteur qui impact directement l'hypervolume, mais qui crée aussi d'autres problèmes. En effet, étant donné que nous utilisons la fonction Chebyshev comme fonction d'agrégation, le point de référence $z^\star$ est indispensable. Or, des travaux récents montrent qu'il est important de rapidement identifier ce point de référence au cours du processus de recherche pour accélérer la convergence [102]. Si le point de référence n'est pas bien défini, les solutions peuvent être attribuées aux mauvais vecteurs de poids. Par exemple sur la Figure 2.6, on peut voir que le point $z^\star$ a été modifié entre la génération 1 et 2. Pour la génération 1, la solution en rouge semble mieux optimiser le sous-problème lié au vecteur W_4. À la génération suivante (la génération 2), alors que le point de référence a été mis à jour, on peut voir que cette même solution optimise dorénavant mieux le

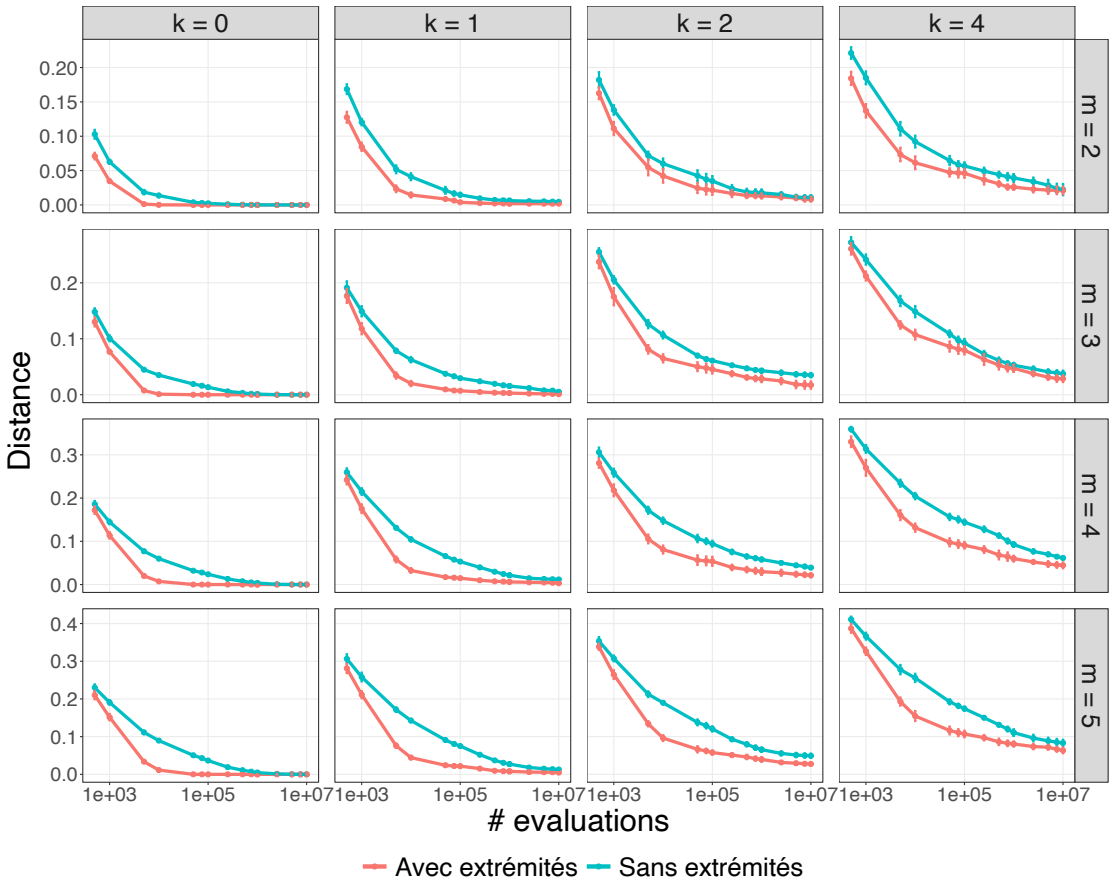

FIGURE 2.5 – Évolution de la distance entre le point de référence optimal de l'instance et le point de référence de l'algorithme au fur et à mesure du nombre de solutions évaluées.

sous-problème avec le vecteur w_2 que celui avec le vecteur w_4. Le vecteur de poids attribué à une solution et les coordonnées du point de références sont des critères qui jouent sur la valeur d'agrégation calculée. Si la valeur d'agrégation calculée n'est pas optimale (calculée avec le vecteur de poids qui convient le mieux et le bon point de référence) alors les étapes de sélection et de remplacement vont être impactées dans le choix des solutions, ce qui va ralentir le processus de recherche.

Le point de référence $z^\star$ est défini grâce aux solutions situées aux extrémités du front Pareto. La Figure 2.5 nous montre l'évolution de la distance entre le point de référence optimal et le point de référence de l'algorithme au cours de son exécution. Pour rappel, étant donné que nous sommes dans un contexte boite-noire, le point de référence doit être mis à jour avec les meilleures valeurs objectif trouvées par les nouvelles solutions générées et évaluées au cours du processus évolutionnaire. On peut

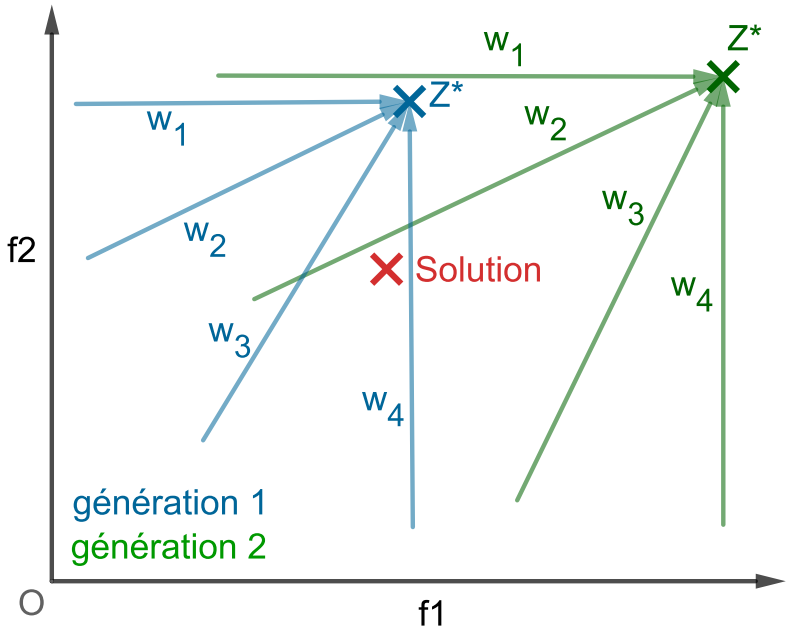

FIGURE 2.6 – Visualisation de la modification du point de référence pour un problème à 2 objectifs.

voir sur la Figure 2.5 que la distance entre le point de référence optimal et le point de référence trouvé par l'algorithme diminue beaucoup plus vite quand l'algorithme sélectionne les sous-problèmes situés aux extrémités.

Cette analyse nous conduit alors à penser que l'inclusion des sous-problèmes situés aux extrémités est nécessaire afin de ne pas ralentir le processus de recherche. Sans ces sous-problèmes, MOEA/D ne peut pas trouver les solutions situées aux extrémités du front Pareto. Ce problème est amplifié par l'utilisation de la fonction d'agrégation Chebyshev, qui ralenti le processus de recherche quand le point de référence optimal n'est pas rapidement atteint. Or, ce point est directement estimé grâce aux solutions se trouvant aux extrémités du front Pareto. Dans la suite de ce chapitre, on ajoutera donc systématiquement les m sous-problèmes situés aux extrémités, en plus des λ sous-problèmes sélectionnés par la stratégie sps.

2.3.4 Étude du nombre de sous-problèmes à sélectionner dans MOEA/D-(μ, λ, sps)

Afin de comparer au mieux les différentes stratégies de sélection des sous-problèmes, nous allons tout d'abord nous intéresser au paramètre λ, c'est-à-dire au nombre de sous-problèmes que va sélectionner la stratégie sps. Pour cela, nous analysons l'impact

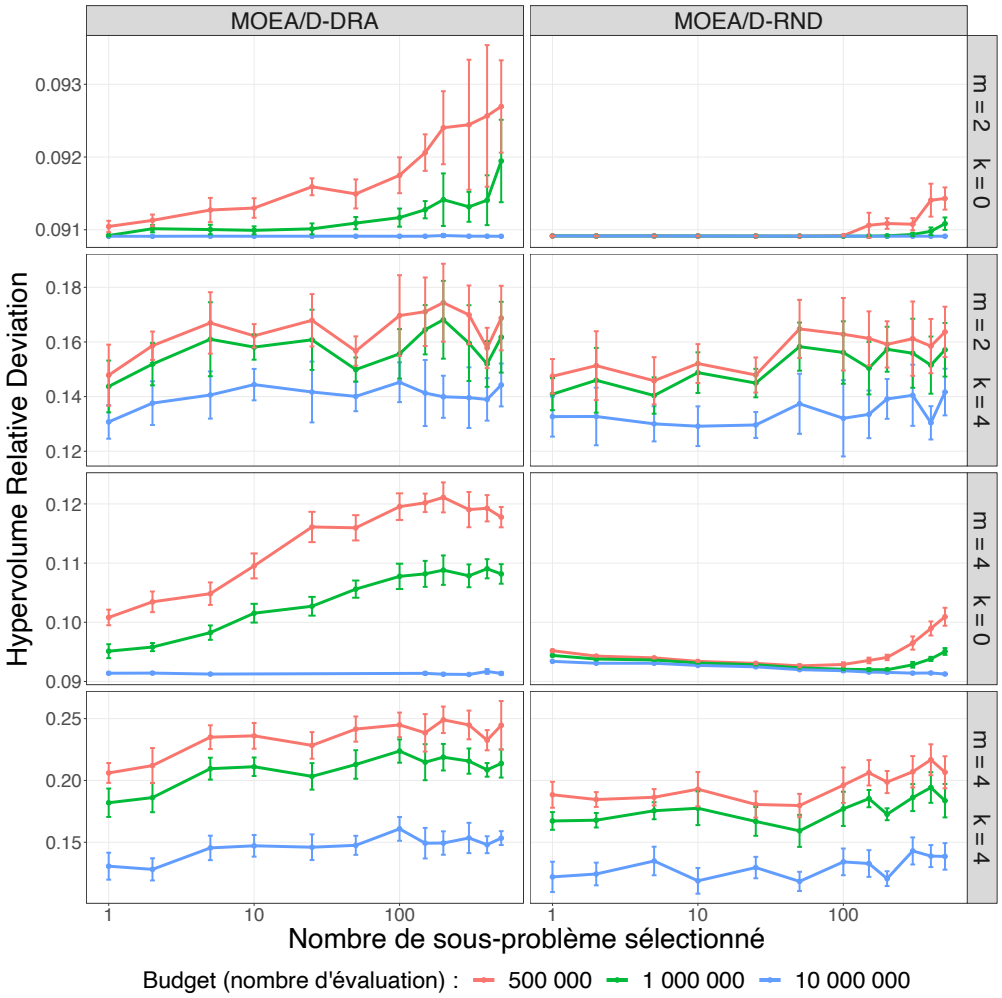

FIGURE 2.7 – Évolution de la qualité du front Pareto en fonction du nombre des sous-problèmes sélectionnés (λ) pour MOEA/D-(μ, λ, sps_{DRA}) et MOEA/D-(μ, λ, sps_{RND}) avec $\mu = 500$

de λ indépendamment de la stratégie *sps*. Nous étudions ainsi l'impact de λ pour chacune des stratégies, en dehors de sps_{ALL} où le nombre de sous-problèmes sélectionnés est fixe : $\lambda = \mu$. Les résultats sont représentés dans la Figure 2.7 pour les stratégies sps_{DRA} et sps_{RND}. La taille de la population est fixée à $\mu = 500$, car d'après nos résultats précédents, une grande population permet d'obtenir des meilleurs résultats sous un budget maximal de 10^7 évaluations. Les différentes valeurs de considérées sont $\lambda \in \{1, 2, 5, 10, 25, 50, 100, 150, 200, 300, 400, 450, 500\}$, que nous étudions pour 3 budgets intermédiaires : 500 000, 1 000 000 et 10 000 000 évaluations. Pour améliorer la visibilité des figures et limiter le nombre d'expériences à réaliser étant donné le nombre important

de valeurs de λ, les résultats sont montrés pour 2 instances linéaires ($k = 0$) avec 2 et 4 objectifs ($m \in \{2, 4\}$) ainsi que 2 instances plus compliquées, avec plus d'optima locaux ($k = 4$), avec 2 et 4 objectifs ($m \in \{2, 4\}$) également.

2.3.4.1 Étude de l'impact de λ pour la stratégie sps_{RND}

Les résultats de la stratégie sps_{RND} sont affichés sur la colonne de droite de la Figure 2.7. Intéressons-nous dans un premier temps aux problèmes linéaires avec peu d'optima locaux, autrement dit les instances de mnk-landscapes avec $k = 0$. Sur ces instances, on peut remarquer qu'un faible nombre de sous-problèmes sélectionnés ($\lambda \leq 100$) donne les meilleurs résultats quand le budget n'est pas élevé. Pour un faible budget, les différences de performance entre $\lambda = 1$ et $\lambda \leq 100$ sont faibles alors que la qualité des résultats décroît rapidement en augmentant la valeur de λ au-delà de 100 sous-problèmes sélectionnés, c'est-à-dire quand nous sélectionnons plus de 1/5 du nombre total de sous-problèmes disponibles. Quand le budget disponible augmente, on peut voir que la valeur de lambda qui donne les meilleurs résultats augmente jusqu'à atteindre la valeur maximale où $\lambda = \mu$. Pour l'instance linéaire à 4 objectifs ($m = 4$ et $k = 0$), on peut très clairement voir que la courbe s'inverse en augmentant la valeur de λ. On le voit beaucoup moins avec la même instance en 2 objectifs ($m = 2$ et $k = 0$) car sps_{RND} semble trouver le front optimal avec une valeur faible de λ et peu de budget, alors qu'il faut plus de 10^6 évaluations pour avoir un résultat similaire en sélectionnant plus de 100 sous-problèmes ($\lambda \geq 100$). Pour résumer, le nombre de sous-problèmes à sélectionner pour des instances avec peu d'optima locaux semble croître en augmentant le budget disponible. Cette tendance ne semble pourtant pas s'appliquer sur des problèmes plus compliqués, quand le paysage devient plus rugueux.

Quand la rugosité du problème augmente, les effets remarqués sur le paramètre λ semblent changer. Si on examine les instances avec le plus grand degré d'épistasie $k = 4$, on remarque que les meilleurs résultats sont obtenus avec un faible nombre de sous-problèmes sélectionnés $\lambda \leq 50$, et ceci indépendamment du budget disponible. Il semble y avoir deux phases de performance en fonction du nombre de sous-problèmes sélectionnés. En effet, les performances semblent similaires quand $1 \leq \lambda \leq 50$ et quand $50 \leq \lambda \leq 500$ et ceci indépendamment du budget. L'identification de réglage optimal de λ optimale semble donc plus compliqué pour des instances plus difficiles. On peut cependant quand même identifier qu'un faible nombre de sous-problèmes sélectionnés aléatoirement donne de meilleurs résultats que de sélectionner tous les sous-problèmes disponibles comme dans MOEA/D-(μ, μ, sps_{ALL}), où $\lambda = \mu = 500$.

Pour conclure sur l'impact du nombre de sous-problèmes sélectionnés par la stratégie sps_{RND}, on peut dire que choisir peu de sous-problèmes, indépendamment de l'instance de mnk-landscapes, et même aléatoirement, permet de se rapprocher plus efficacement du front Pareto que MOEA/D-(μ, μ, sps_{ALL}). Plus le problème est rugueux et plus le bénéfice apporté par une faible valeur de λ semble s'atténuer pour la stratégie sps_{RND}. On peut expliquer cela en se rappelant que le nombre de sous-problèmes sélectionnés correspond également au nombre de solutions générées lors de chaque génération. Si le problème est complexe, il est plus rare d'améliorer les différents sous-problèmes ou la population. Mais si le problème est plus facile, on peut alors penser que générer peu de solutions améliorera tout aussi vite la population que si on en avait généré beaucoup. Le fait de générer, et donc d'évaluer moins de solutions permet de passer à la génération suivante plus rapidement sans perdre de ressources de calculs inutilement.

2.3.4.2 Étude de l'impact de λ pour la stratégie sps_{DRA}

Les résultats de la stratégie sps_{DRA} sont présentés sur la colonne de gauche de la Figure 2.7. Les résultats obtenus pour la stratégie sps_{DRA} semblent beaucoup plus homogènes que pour la stratégie sps_{RND}. En effet, l'impact du nombre de sous-problèmes sélectionnés semble moins sensible au budget disponible et à la difficulté du problème. Plus précisément, on peut voir que peu importe le budget disponible ou l'instance de mnk-landscapes, les meilleures performances de la stratégie sps_{DRA} sont obtenues avec la valeur $\lambda = 1$. On peut voir que plus le nombre de sous-problèmes sélectionnés est important, plus la qualité des résultats se détériore, et ceci indépendamment du budget. Ce phénomène peut s'expliquer ici par la nature même de la stratégie sps_{DRA}. En effet, cette stratégie va sélectionner les sous-problèmes en fonction de leur amélioration relative. Le fait que cette stratégie soit plus "intelligente" que la stratégie sps_{RND} permet alors de diminuer la probabilité de choisir des sous-problèmes moins intéressants à optimiser, en diminuant le nombre de sous-problèmes sélectionnés. On peut noter toutefois que comme pour la stratégie sps_{RND}, ce phénomène semble s'atténuer quand on augmente la difficulté du problème. En augmentant la difficulté du problème, que ça soit en augmentant sa rugosité ou son nombre d'objectifs, ce paramètre semble donc plus compliqué à configurer.

On peut également noter que dans la version initiale de MOEA/D–DRA [117], le nombre de sous-problèmes sélectionnés était toujours de $\mu/5$. Avec ce même paramétrage, on choisirait alors ici $\lambda = 100$. On remarque alors que cette valeur n'est pas du tout optimale pour notre classe de problème, mais également qu'elle peut être améliorée

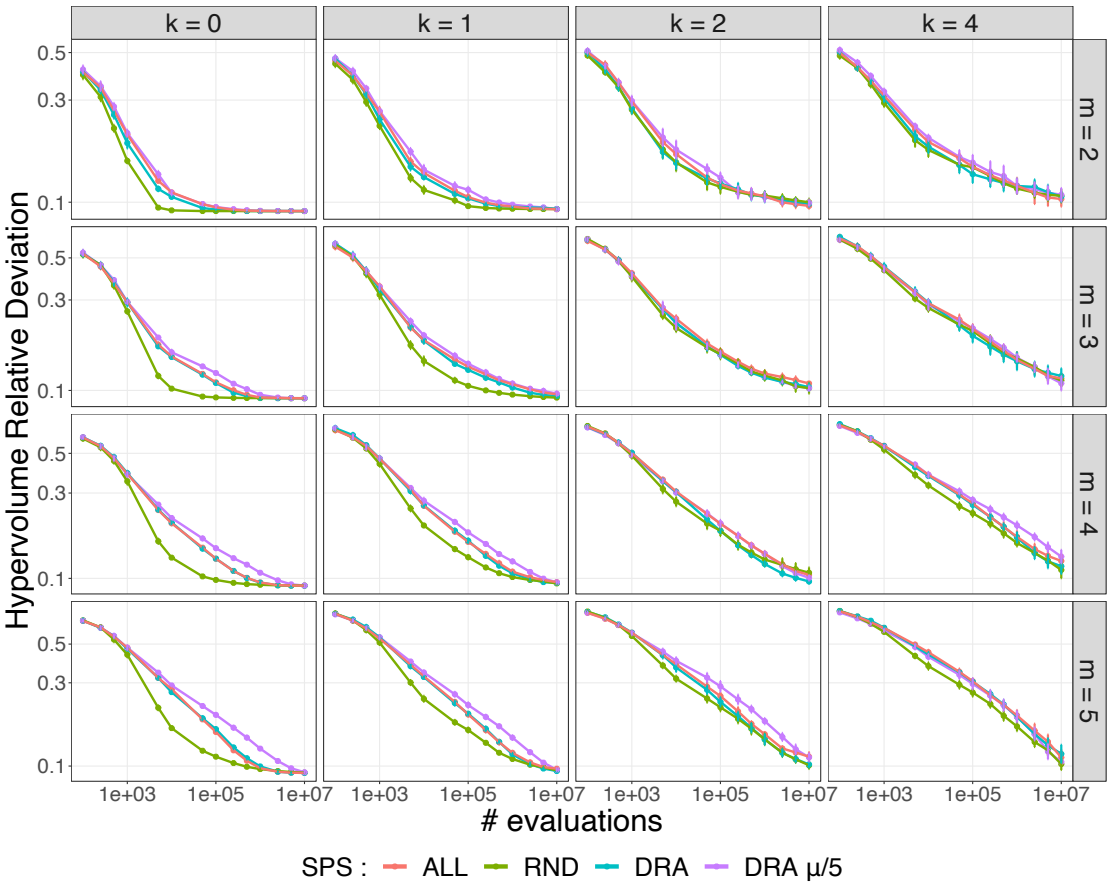

FIGURE 2.8 – Profils de convergence de MOEA/D-(μ, λ, *sps*) en fonction du nombre de solutions dans sa population avec $\mu = 500$.

simplement avec $\lambda = 1$, qui améliore considérablement les performances sur les quatre instances considérées dans la Figure 2.7.

2.3.5 Étude des stratégies de sélection des sous-problèmes dans MOEA/D-(μ, λ, *sps*)

Maintenant que nous avons pu identifier des valeurs convenables de λ pour chacune des stratégies de sélection des sous-problèmes *sps* étudiées, nous pouvons comparer ces différentes stratégies en utilisant les valeurs optimales de λ pour chacune d'entre elles. La Figure 2.8 montre ainsi les profils de convergence des configurations suivantes : MOEA/D-($\mu = 500$, $\lambda = \mu$, sps_{ALL}), MOEA/D-($\mu = 500$, $\lambda = 1$, sps_{RND}), MOEA/D-($\mu = 500$, $\lambda = 1$, sps_{DRA}), MOEA/D-($\mu = 500$, $\lambda = \mu/5$, sps_{DRA}). La valeur de $\lambda = 1$ a été choisie pour les stratégies sps_{RND} et sps_{DRA}. Cette valeur montre les résultats les

plus convaincants pour sps_{DRA} sur la majorité des instances étudiées. Pour la stratégie sps_{RND}, même si d'autres valeurs de λ peuvent s'avérer légèrement plus performantes, choisir $\lambda = 1$ donne tout de même de bons résultats et nous permet de comparer les stratégies sps_{DRA} et sps_{RND} avec le même paramétrage. Étant donné que nous n'avons pas pu expérimenter toutes les valeurs de λ pour les différentes stratégies sur toutes les instances de mnk-landscapes, nous ne pouvons affirmer que $\lambda = 1$ est la meilleure valeur de λ pour sps_{DRA} sur les instances combinatoires de mnk-landscapes. Nous allons par conséquent également étudier la valeur par défaut pour la stratégie sps_{DRA}, qui est utilisée dans l'article original [117], c'est-à-dire $\lambda = \mu/5$. Pour rappel, comme mis en évidence dans la section précédente, dès que nous choisissons un sous-ensemble de solutions à parcourir au lieu de parcourir toute la population, nous ajoutons aux λ sous-problèmes sélectionnés par la stratégie *sps*, les m sous-problèmes situés aux extrémités.

On peut voir dans la Figure 2.8 que la stratégie sps_{RND} a un profil de convergence bien supérieur aux autres stratégies, indépendamment des instances étudiées. Le fait de sélectionner un sous-problème aléatoirement (ainsi que les sous-problèmes situés aux extrémités) permet de s'approcher beaucoup plus efficacement du front Pareto que les stratégies sps_{ALL} et sps_{DRA}. Les résultats sont plus évidents sur les instances avec peu d'optima locaux, c'est-à-dire dans notre étude les instances de mnk-landscapes avec $k \leq 1$. Quand on augmente le degré d'épistasie, on peut voir que les résultats se rapprochent de ceux de sps_{DRA}, mais restent tout de même supérieurs à la stratégie sps_{ALL}. Néanmoins, bien que la stratégie sps_{RND} semble se rapprocher très rapidement du front Pareto, elle semble aussi présenter quelques faiblesses quand le budget est particulièrement élevé. On peut le voir en particulier sur les instances les plus simples si on regarde plus en détail le Tableau 2.2. Ce tableau affiche le rang de chaque stratégie en fonction de l'instance et du budget étudié. Un rang c indique que la stratégie correspondante a été surpassée par c autres stratégies selon un test statistique de Wilcoxon avec un niveau de conformité de 0.05. La valeur entre parenthèse correspond à la déviation relative moyenne d'hypervolume. Lorsque cette valeur est soulignée, cela signifie qu'il s'agit de la meilleure valeur pour une instance et un budget donné. On peut voir sur ce tableau que la stratégie sps_{RND} a quelques difficultés sur les instances les plus simples, pour lesquelles on atteint rapidement le front Pareto avec la version classique de MOEA/D (instances de mnk-landscapes avec $k = 0$). En effet, la stratégie sps_{RND} ne semble pas en mesure de couvrir correctement l'entièreté du front Pareto, et présente un rang de 2 lorsque le budget est de 10^7 évaluations. Néanmoins, elle semble s'y approcher bien plus rapidement que les autres stratégies, avec un rang à 0, lorsque le budget est de 10^4,

TABLEAU 2.2 – Rangs et moyenne (entre parenthèse) de la déviation relative d'hypervolume, obtenue par les différentes stratégies *sps* après $\left\{10^4, 10^5, 10^6, 10^7\right\}$ évaluations (une petite valeur est meilleure). Pour les stratégies sps_{DRA} et sps_{RND}, $\lambda = 1$. Pour chaque budget, un rang c indique que la stratégie correspondante a été surpassée par c autres stratégies selon un test statistique de Wilcoxon avec un niveau de conformité de 0.05. Les rangs en gras signifient que les stratégies correspondantes ne se sont pas significativement surpassées, les valeurs soulignées correspondent à la meilleure valeur moyenne.

		10^4 evaluations			10^5 evaluations			10^6 evaluations			10^7 evaluations		
m	k	sps_{ALL}	sps_{DRA}	sps_{RND}	sps_{ALL}	sps_{DRA}	sps_{RND}	sps_{ALL}	sps_{DRA}	sps_{RND}	sps_{ALL}	sps_{DRA}	sps_{RND}
2	0	$2_{(11.2)}$	$1_{(10.5)}$	$\mathbf{0}_{(\underline{09.1})}$	$2_{(09.4)}$	$1_{(09.3)}$	$\mathbf{0}_{(\underline{09.0})}$	$2_{(09.1)}$	$\mathbf{0}_{(09.0)}$	$\mathbf{0}_{(\underline{09.0})}$	$\mathbf{0}_{(09.0)}$	$\mathbf{0}_{(\underline{09.0})}$	$2_{(09.0)}$
	1	$1_{(14.0)}$	$1_{(13.2)}$	$\mathbf{0}_{(\underline{11.4})}$	$1_{(10.9)}$	$1_{(10.5)}$	$\mathbf{0}_{(\underline{09.5})}$	$2_{(09.8)}$	$1_{(09.6)}$	$\mathbf{0}_{(\underline{09.2})}$	$2_{(09.5)}$	$1_{(09.4)}$	$\mathbf{0}_{(\underline{09.2})}$
	2	$1_{(17.2)}$	$\mathbf{0}_{(15.6)}$	$\mathbf{0}_{(\underline{15.4})}$	$1_{(13.0)}$	$\mathbf{0}_{(12.5)}$	$\mathbf{0}_{(\underline{11.7})}$	$\mathbf{0}_{(11.3)}$	$\mathbf{0}_{(10.9)}$	$\mathbf{0}_{(\underline{10.6})}$	$\mathbf{0}_{(10.2)}$	$\mathbf{0}_{(10.1)}$	$\mathbf{0}_{(\underline{09.8})}$
	4	$2_{(22.1)}$	$\mathbf{0}_{(19.5)}$	$\mathbf{0}_{(\underline{18.9})}$	$1_{(17.5)}$	$\mathbf{0}_{(\underline{14.9})}$	$\mathbf{0}_{(16.0)}$	$2_{(14.6)}$	$\mathbf{0}_{(13.3)}$	$\mathbf{0}_{(\underline{13.0})}$	$\mathbf{0}_{(13.1)}$	$\mathbf{0}_{(\underline{12.0})}$	$\mathbf{0}_{(12.2)}$
3	0	$1_{(15.1)}$	$1_{(15.0)}$	$\mathbf{0}_{(\underline{10.2})}$	$1_{(11.0)}$	$1_{(10.9)}$	$\mathbf{0}_{(\underline{09.2})}$	$\mathbf{0}_{(09.1)}$	$\mathbf{0}_{(09.1)}$	$\mathbf{0}_{(\underline{09.1})}$	$\mathbf{0}_{(\underline{09.0})}$	$\mathbf{0}_{(09.0)}$	$2_{(09.1)}$
	1	$1_{(18.4)}$	$1_{(18.4)}$	$\mathbf{0}_{(\underline{14.2})}$	$1_{(13.3)}$	$1_{(13.0)}$	$\mathbf{0}_{(\underline{10.5})}$	$1_{(10.8)}$	$1_{(10.5)}$	$\mathbf{0}_{(\underline{09.4})}$	$1_{(09.5)}$	$1_{(09.5)}$	$\mathbf{0}_{(\underline{09.0})}$
	2	$1_{(24.4)}$	$1_{(23.4)}$	$\mathbf{0}_{(\underline{20.6})}$	$1_{(16.3)}$	$\mathbf{0}_{(16.1)}$	$\mathbf{0}_{(\underline{14.7})}$	$2_{(12.8)}$	$1_{(12.3)}$	$\mathbf{0}_{(\underline{11.1})}$	$1_{(11.7)}$	$1_{(11.1)}$	$\mathbf{0}_{(\underline{09.4})}$
	4	$1_{(31.0)}$	$\mathbf{0}_{(29.9)}$	$\mathbf{0}_{(\underline{28.0})}$	$\mathbf{0}_{(22.7)}$	$\mathbf{0}_{(\underline{20.5})}$	$\mathbf{0}_{(21.4)}$	$\mathbf{0}_{(16.7)}$	$\mathbf{0}_{(\underline{15.4})}$	$\mathbf{0}_{(15.6)}$	$\mathbf{0}_{(13.6)}$	$\mathbf{0}_{(13.0)}$	$\mathbf{0}_{(\underline{12.2})}$
4	0	$1_{(20.5)}$	$1_{(20.5)}$	$\mathbf{0}_{(\underline{13.2})}$	$1_{(12.9)}$	$1_{(12.8)}$	$\mathbf{0}_{(\underline{09.9})}$	$\mathbf{0}_{(09.4)}$	$\mathbf{0}_{(09.4)}$	$\mathbf{0}_{(\underline{09.3})}$	$\mathbf{0}_{(\underline{09.0})}$	$\mathbf{0}_{(09.0)}$	$2_{(09.2)}$
	1	$1_{(25.0)}$	$1_{(24.9)}$	$\mathbf{0}_{(\underline{18.5})}$	$1_{(15.2)}$	$1_{(15.4)}$	$\mathbf{0}_{(\underline{11.8})}$	$1_{(09.9)}$	$1_{(09.8)}$	$\mathbf{0}_{(\underline{08.8})}$	$1_{(08.4)}$	$1_{(08.5)}$	$\mathbf{0}_{(\underline{07.9})}$
	2	$1_{(29.8)}$	$1_{(30.6)}$	$\mathbf{0}_{(\underline{24.7})}$	$1_{(19.5)}$	$1_{(18.7)}$	$\mathbf{0}_{(\underline{16.0})}$	$1_{(12.9)}$	$1_{(12.3)}$	$\mathbf{0}_{(\underline{10.1})}$	$1_{(09.8)}$	$1_{(09.9)}$	$\mathbf{0}_{(\underline{07.9})}$
	4	$1_{(38.0)}$	$1_{(37.1)}$	$\mathbf{0}_{(\underline{32.5})}$	$1_{(26.4)}$	$1_{(25.4)}$	$\mathbf{0}_{(\underline{22.5})}$	$1_{(16.8)}$	$1_{(16.6)}$	$\mathbf{0}_{(\underline{15.1})}$	$\mathbf{0}_{(11.6)}$	$\mathbf{0}_{(11.3)}$	$\mathbf{0}_{(\underline{10.5})}$
5	0	$2_{(26.3)}$	$1_{(25.2)}$	$\mathbf{0}_{(\underline{15.4})}$	$1_{(14.1)}$	$1_{(14.7)}$	$\mathbf{0}_{(\underline{10.2})}$	$\mathbf{0}_{(\underline{08.0})}$	$1_{(08.3)}$	$2_{(08.5)}$	$\mathbf{0}_{(\underline{07.4})}$	$\mathbf{0}_{(07.5)}$	$2_{(08.0)}$
	1	$1_{(29.9)}$	$1_{(29.9)}$	$\mathbf{0}_{(\underline{21.5})}$	$1_{(16.5)}$	$1_{(17.1)}$	$\mathbf{0}_{(\underline{13.0})}$	$1_{(08.3)}$	$1_{(08.5)}$	$\mathbf{0}_{(\underline{07.7})}$	$\mathbf{0}_{(\underline{05.9})}$	$\mathbf{0}_{(06.1)}$	$1_{(06.1)}$
	2	$1_{(35.2)}$	$1_{(34.1)}$	$\mathbf{0}_{(\underline{28.1})}$	$1_{(21.4)}$	$\mathbf{0}_{(20.0)}$	$\mathbf{0}_{(\underline{17.6})}$	$\mathbf{0}_{(10.9)}$	$\mathbf{0}_{(10.5)}$	$\mathbf{0}_{(\underline{09.7})}$	$\mathbf{0}_{(06.8)}$	$\mathbf{0}_{(06.2)}$	$\mathbf{0}_{(\underline{05.3})}$
	4	$1_{(41.6)}$	$1_{(40.7)}$	$\mathbf{0}_{(\underline{35.5})}$	$1_{(26.5)}$	$1_{(26.9)}$	$\mathbf{0}_{(\underline{24.1})}$	$\mathbf{0}_{(14.7)}$	$\mathbf{0}_{(15.1)}$	$\mathbf{0}_{(\underline{14.3})}$	$\mathbf{0}_{(\underline{06.0})}$	$\mathbf{0}_{(07.4)}$	$\mathbf{0}_{(07.4)}$

10^5 ou 10^6 évaluations.

Même si la stratégie sps_{DRA} obtient globalement de meilleurs résultats que sps_{ALL}, on peut voir très clairement que la fonction *d'amélioration relative* utilisée pour détecter les sous-problèmes difficiles semble amoindrir les performances. Le fait que sps_{RND} soit bien meilleure que sps_{DRA} dans toute la phase d'approche vers le front Pareto montre que la stratégie sps_{DRA} ne choisit pas les bons sous-problèmes, car elle ne détecte pas la présence de sous-problèmes plus difficiles que d'autres. Avec ce constat, on peut remarquer que l'utilisation d'une stratégie d'allocation, même des plus basique comme sps_{RND} améliore tout de même les performances de MOEA/D. La restructuration du parcours des sous-problèmes semble donc être le point important derrière le système d'allocation des efforts de calcul proposé dans [117], avant même la stratégie de sélection de ces sous-problèmes.

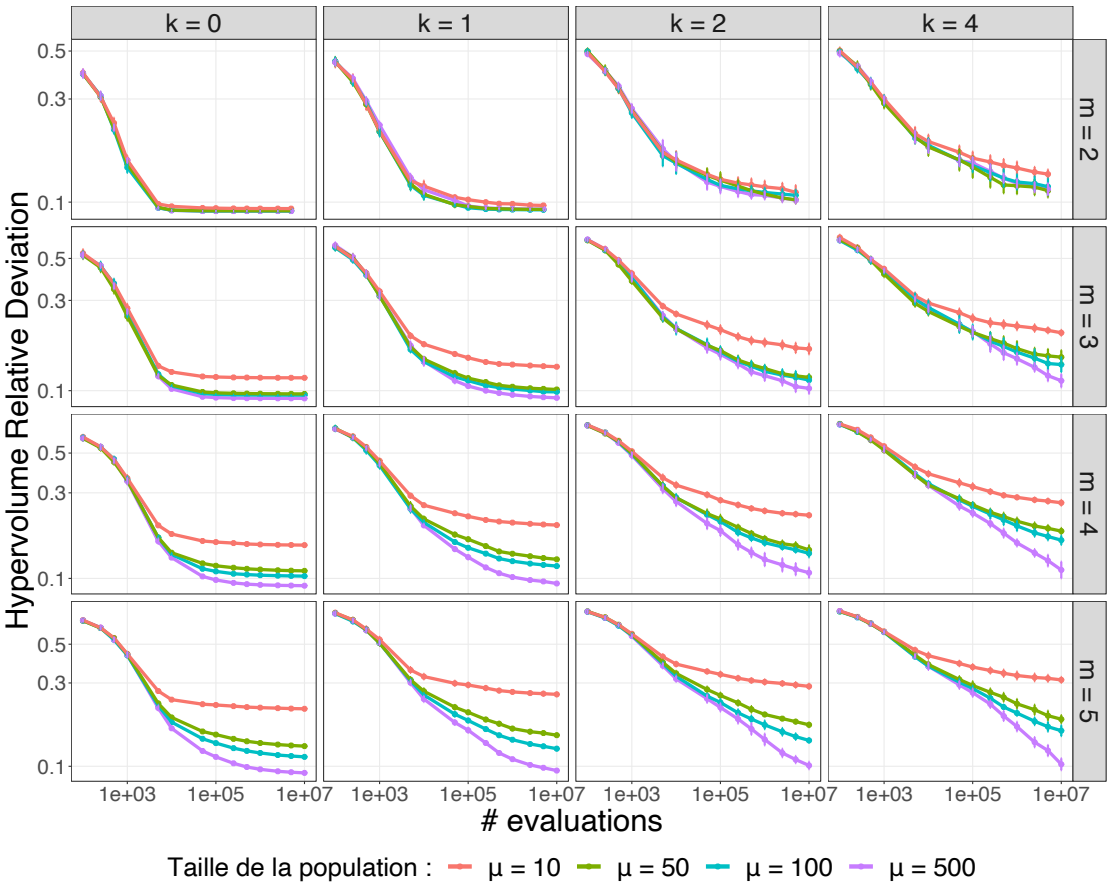

FIGURE 2.9 – Profils de convergence de MOEA/D-(μ, λ, sps_{RND}) en fonction du nombre de solutions dans sa population.

2.3.6 Étude de la taille de population dans MOEA/D-(μ, λ, sps)

Maintenant que les paramètres λ et sps ont été étudiés dans la section précédente, nous allons pouvoir analyser l'impact du paramètre μ. Dans le framework MOEA/D-(μ, λ, sps), ce paramètre a le rôle unique de représenter le nombre de solutions présentes dans la population. Contrairement à l'étude dans la Section 2.3.2 sur MOEA/D, autrement écrit MOEA/D-(μ, μ, sps_{ALL}), nous étudierons ici ce paramètre indépendamment du nombre de solutions λ générées à chaque génération.

Suite à notre étude dans la Section 2.3.2, nous avons fixé la taille de la population à $\mu = 500$, étant donné qu'il s'agissait de la valeur qui couvrait le mieux l'entièreté du front Pareto lorsque le budget était suffisamment important. Cette valeur n'était cependant pas optimale pour des budgets moins élevés, où les plus petites populations donnaient

de meilleurs résultats. Avec l'utilisation d'un système d'allocation de ressources comme dans MOEA/D–DRA, nous espérions pouvoir combler les faiblesses des grandes populations dans toute la phase d'approche du front Pareto. La stratégie $\mathsf{sps}_{\mathrm{RND}}$ nous a prouvé que cela était possible, même si celle-ci n'identifie pas de sous-problèmes plus difficiles, pour lesquels il aurait fallu concentrer plus de ressources. Nous allons alors compléter nos analyses précédentes en étudiant avec la Figure 2.9 l'impact du nombre de solutions dans la population pour l'algorithme MOEA/D-(μ, λ, $\mathsf{sps}_{\mathrm{RND}}$), toujours en ne sélectionnant qu'un seul sous-problème aléatoirement ($\lambda = 1$), en plus des m sous-problèmes situés aux extrémités.

Contrairement aux résultats obtenus par MOEA/D-(μ, μ, $\mathsf{sps}_{\mathrm{ALL}}$), reportés sur la Figure 2.2, on peut voir ici sur la Figure 2.9 que les profils de convergence de MOEA/D-(μ, λ, *sps*) avec $\lambda = 1$ sont beaucoup plus stables sur l'ensemble des instances considérées. En effet, contrairement à MOEA/D-(μ, μ, $\mathsf{sps}_{\mathrm{ALL}}$), on peut voir ici que MOEA/D-(μ, λ, $\mathsf{sps}_{\mathrm{RND}}$) obtient de meilleures performances lorsque la taille de la population μ est plus élevée, et cela, peu importe le budget disponible. On peut toutefois noter que la définition de "grande" population reste toujours à définir en fonction de l'instance de problème à optimiser. Pour les problèmes les plus simples, augmenter la taille de la population n'apporte pas d'amélioration. Par exemple pour les instances à 2 objectifs ($m = 2$) avec un degré d'épistasie assez faible ($k \leq 1$), nous n'observons aucune différence lorsque la population possède plus de 50 solutions.

2.4 Conclusion

Les travaux présentés dans ce chapitre nous ont permis de revoir la conception générale de MOEA/D en apportant une reformulation de haut niveau, mais plus précise, du schéma de stratégie évolutionnaire ($\mu + \lambda$) inhérent au framework. Nous avons ainsi analysé le rôle des trois composants contrôlant la gestion des efforts de calculs au sein de l'algorithme : la taille de la population (μ), le nombre de sous-problèmes sélectionnés à chaque génération (λ), et la stratégie utilisée pour sélectionner les λ sous-problèmes (*sps*). Notre analyse de la taille de la population dans la première version de MOEA/D a montré que les performances de l'algorithme peuvent être améliorées en augmentant le nombre de solutions dans la population. Notre étude sur les effets combinés de ces paramètres a confirmé notre première analyse, en montrant que la plus simple des stratégies de sélection des sous-problèmes permet d'améliorer les performances de MOEA/D.

Nous aimerions également noter que les travaux de la littérature portant sur ce

thème sont souvent focalisés sur la sélection des sous-problèmes les plus difficiles afin d'optimiser les efforts de calcul au sein même de MOEA/D. À travers nos travaux, nous avons ainsi pu mettre en évidence que l'optimisation des efforts de calculs dans MOEA/D peut se faire facilement à travers d'autres composants. Ainsi, le nombre de sous-problèmes sélectionnés, qui correspond également au nombre de solutions enfant générées par génération, semble dans notre étude au moins aussi importante que la stratégie qui sélectionne ces sous-problèmes.

Nous sommes maintenant en mesure de donner une recommandation du paramétrage de MOEA/D-(μ, λ, *sps*) en fonction des propriétés générales du problème. Cependant, dans un contexte d'optimisation boîte-noire, bien que le nombre d'objectifs puisse être connu avant le processus d'optimisation, d'autres propriétés ne le sont pas, comme la rugosité (k) par exemple. De ce fait, plusieurs perspectives pour étendre notre analyse peuvent être envisagées, autant en optimisation combinatoire qu'en optimisation continue. Une direction de recherche intéressante consisterait à déduire les propriétés du paysage avec les informations récoltées au cours du processus de recherche pour apprendre le meilleur paramétrage de μ, λ et *sps*, que ce soit hors ligne (avant l'exécution de l'algorithme) ou en ligne (pendant l'exécution de l'algorithme). Ceci permettrait non seulement d'éviter un effort de configuration pour l'utilisateur, mais pourrait également amener à un meilleur comportement de l'algorithme au fur et à mesure de son déroulement. On pourrait par exemple considérer un réglage adaptatif de μ, λ et *sps* en ajustant leurs valeurs en fonction du comportement observé sur les différentes générations réalisées. Nos travaux ont également permis de mieux mettre en évidence l'importance des sous-problèmes situés aux extrémités, notamment quand on ne sélectionne pas tous les sous-problèmes à chaque génération. Nous avons pris le choix dans notre étude de les sélectionner systématiquement pour mieux se comparer aux algorithmes état-de-l'art tels que MOEA/D–DRA. Il serait intéressant d'étudier une sélection obligatoire au début du processus de recherche uniquement. Selon notre étude, ces sous-problèmes sont impactant pour trouver rapidement le point de référence, il semble donc inutile de les sélectionner une fois ce point identifié.

Chapitre 3

Mécanismes d'échappement dans MOEA/D

Résumé

Dans ce troisième chapitre, nous présentons la deuxième contribution de cette thèse, dans laquelle nous nous intéressons à l'intégration de mécanismes d'échappement dans le framework MOEA/D. En effet, concevoir des mécanismes permettant d'éviter d'être coincé dans certaines zones de l'espace de recherche est un aspect fondamental pour la réussite de tout algorithme évolutionnaire. L'idée générale développée dans ce chapitre est basée sur une observation simple : grâce à sa nature basée sur la décomposition d'un problème multi-objectif en plusieurs sous-problèmes mono-objectifs, le framework MOEA/D permet d'intégrer des techniques d'échappement issues de l'optimisation mono-objectif pour l'optimisation multi-objectif. Ainsi, nous montrons que nous pouvons étendre et étudier de façon simple et unifiée des mécanismes d'échappement jusque-là largement adoptés dans la communauté mono-objectif dans le contexte de l'optimisation multi-objectif. Ceci est rendu possible d'abord en agrégeant les observations faites localement au niveau des différents sous-problèmes définis dans MOEA/D pour détecter le moment où le processus de recherche multi-objectif est coincé, et ensuite en étudiant des mécanismes standard pour permettre à la population d'évoluer vers de nouvelles zones de l'espace de recherche. Une partie de ces travaux a fait l'objet d'une publication au congrès international IEEE CEC 2021 [23].

Dans ce troisième chapitre, nous commençons par introduire le contexte et les motivations liées à l'échappement des optima locaux dans la Section 3.1. Nous y présentons quelques notions liées à l'optimisation mono-objectif, pour ensuite mieux introduire

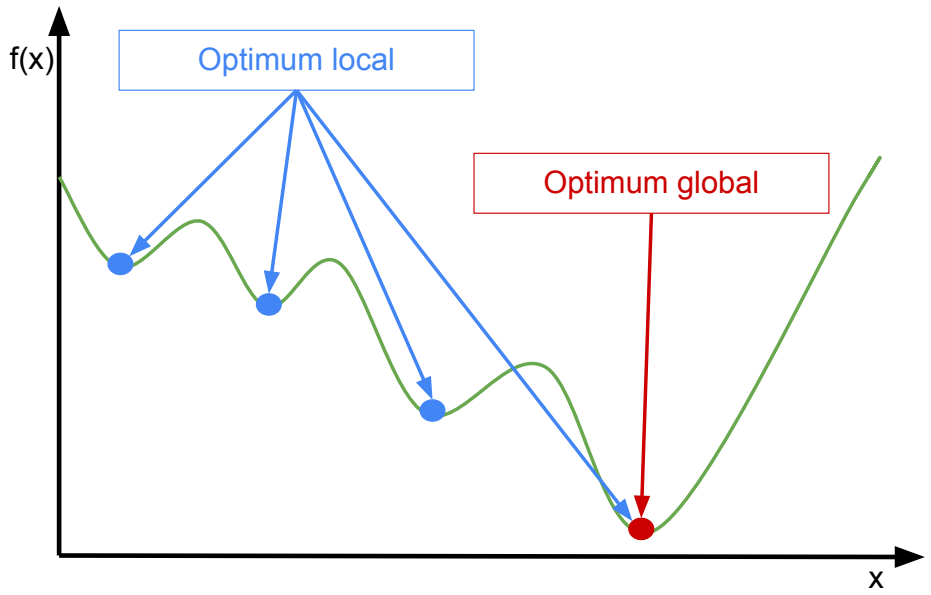

FIGURE 3.1 – Représentation de l'optimalité locale et globale dans un contexte de minimisation de la fonction f.

ces notions en optimisation multi-objectif. Nous proposons ensuite dans la Section 3.2 une intégration d'un mécanisme d'échappement aux optima locaux au sein même de MOEA/D. Nous y présentons le système permettant de les détecter, ainsi que différentes stratégies considérées pour s'y échapper. Ces mécanismes sont ensuite étudiés en détail dans notre analyse expérimentale dans la Section 3.3, avant de conclure ce chapitre dans la Section 3.4.

3.1 Contexte et motivations

De façon générale, et indépendamment du fait que l'on considère un problème mono- ou multi-objectif, un aspect majeur pour la conception d'un algorithme évolutionnaire efficace est celui d'éviter que le processus de recherche ne soit bloqué dans une région sous-optimale de l'espace de recherche. L'idée sous-jacente est de permettre au processus de recherche de s'échapper de ce qu'on appelle les optima locaux, et ainsi d'explorer de nouvelles régions où des solutions de meilleure qualité peuvent potentiellement être trouvées. Pour mieux comprendre cette idée, nous commençons d'abord par rappeler et discuter quelques notions de bases en optimisation mono-objectif, avant de se concentrer sur le contexte de la thèse qui est l'optimisation multi-objectif par décomposition.

3.1.1 Optima locaux en optimisation mono-objectif

En optimisation mono-objectif, on définit habituellement un optimum local de façon assez naturelle en se basant sur : (1) une fonction de fitness (qui à une solution de l'espace de recherche associe une valeur réelle, le plus souvent correspondant à la fonction objectif) et (2) une relation de voisinage entre les différentes solutions de l'espace de recherche. La fonction de fitness permet de mesurer la qualité relative d'une solution et la relation de voisinage permet de définir l'ensemble des solutions proches que l'on peut atteindre à partir d'une solution courante, appelé par abus de langage le voisinage d'une solution donnée. À titre d'exemple, pour des solutions avec une représentation binaire, un voisinage possible, et couramment utilisé, est celui basé sur la distance de Hamming [32], c'est-à-dire que le voisinage d'une chaîne binaire est constitué de toutes les chaînes binaires à une distance de Hamming de 1. On peut ainsi dire de façon plus formelle qu'une solution est un optimum local si elle ne possède aucune solution dans son voisinage ayant une meilleure valeur de fitness. Ainsi, un algorithme simple par *recherche locale* procède de la façon suivante : à partir d'une solution initiale, examiner son voisinage, déterminer s'il existe une solution voisine qui améliore la solution courante, sélectionner un voisin améliorant et itérer ce processus, jusqu'à ce qu'aucune amélioration ne soit possible. Ce type de recherche termine donc dans un optimum local. Il est bien sûr évident que le but ultime de tout algorithme est de trouver un optimum local qui soit aussi un optimum global. Pour cela, la plupart des algorithmes se basant sur des recherches locales intègrent dans leur conception même des mécanismes d'échappement, permettant d'éviter que la recherche ne soit bloquée dans des optima locaux particuliers.

De façon similaire, dans le cadre plus général d'un algorithme évolutionnaire considérant des opérateurs spécifiques pour la création de nouvelles solutions (éventuellement de façon stochastique), comme des opérateurs de croisement, on pourrait considérer que le processus de recherche est bloqué dans un optimum local quand il n'arrive plus à améliorer la ou les solutions courantes [1]. En utilisant la métaphore usuelle en optimisation évolutionnaire, on peut se représenter la structure de l'espace de recherche comme un paysage, dans lequel les optima locaux correspondent à des collines séparées par des vallées qu'il faut parcourir avant de trouver une solution de haute qualité. Le but étant de trouver l'optimum global, c'est-à-dire la plus haute colline du paysage. Les concepts d'optimum local et global sont représentés sur la Figure 3.1 dans un contexte

1. La définition rigoureuse et formelle d'un optimum local est plus complexe dans le cadre général des algorithmes évolutionnaires. Nous adoptons ce point de vue uniquement pour mieux illustrer notre démarche.

Algorithme 3 : Iterated Local Search (ILS)

```
   Input :
   /* génération d'une solution initiale x                          */
1  x ← recherche_locale();
2  x_best ← x;
3  tant que que le critère d'arrêt n'est pas atteint faire
       /* perturbation de la solution x                             */
4      x' ← perturbation(x);
       /* Nouvelle recherche locale en partant de la solution
          perturbée x'                                              */
5      x ← recherche_locale(x');
6      si x est meilleure que x_best alors
7          x_best ← x;
8  retourner x_best
```

de minimisation d'une fonction f.

3.1.2 Stratégies d'échappement en optimisation mono-objectif

En optimisation combinatoire mono-objectif, s'échapper des optima locaux est l'un des sujets les plus étudiés dans la communauté des algorithmes évolutionnaires. Dans le cadre des algorithmes à base de recherche locale, la plupart des stratégies existantes se basent sur l'idée d'autoriser une dégradation ponctuelle des solutions courantes afin de permettre au processus de recherche de repartir vers un nouvel optimum local. En utilisant la métaphore issue des paysages de fitness, qui consiste à vouloir atteindre la plus haute colline (l'optimum global), l'idée est de parcourir le paysage éventuellement rugueux en faisant des "sauts" permettant de passer successivement par plusieurs collines (les optima locaux). On peut ainsi se représenter cela par le fait d'autoriser au processus de recherche de dégrader ses solutions pour redescendre vers une vallée proche et repartir par la suite vers une nouvelle colline. Ci-dessous, nous rappelons quelques stratégies d'échappement couramment utilisées en optimisation mono-objectif dans le cadre de l'algorithme de recherche locale itérée. En effet, les travaux présentés dans ce chapitre peuvent être vus comme une adaptation de ces stratégies dans un contexte multi-objectif.

Une recherche locale itérée, ou ILS (Iterated Local Search) [66], est l'extension naturelle d'une recherche locale simple. Le déroulement d'une ILS basique est résumé dans le template de haut niveau de l'algorithme 3. En effet, comme discuté précédemment,

une recherche locale simple ou *hill-climbing*, c'est-à-dire une recherche par améliorations successives dans le voisinage de la solution courante, ne possède aucun mécanisme capable d'éviter les optima locaux. Autrement dit, la recherche s'arrête obligatoirement sur un optimum local. L'idée générale d'une ILS est de démarrer une nouvelle recherche locale simple, en commençant depuis une nouvelle solution. En pratique, une des questions cruciales dans une ILS est la conception de la stratégie pour la génération de la nouvelle solution initiale à partir de laquelle une nouvelle recherche locale est appliquée. Nous rappelons ci-après deux types de stratégies souvent étudiées dans la littérature :

— *Génération aléatoire d'une solution initiale.* Il s'agit ici de repartir d'une solution initiale générée de façon aléatoire sans lien apparent avec l'optimum local courant. Ce type de stratégie accentue le pouvoir de diversification des optima locaux trouvés. En particulier, ceci permet a priori de minimiser les chances de retrouver le même optimum local que la recherche locale précédente. Cependant, il n'est pas clair en général que tous ces optima locaux aient une qualité acceptable.

— *Perturbation d'un optimum local.* Il s'agit ici d'effectuer une perturbation d'un optimum local afin de s'en éloigner sans pour autant repartir d'une solution complètement aléatoire. En effet, la perturbation consiste en général à modifier les valeurs de quelques variables, ou par exemple à prendre de façon aléatoire une solution dans le voisinage immédiat d'un optimum local. On parle alors de degré de perturbation pour désigner le niveau de similarité entre l'optimum local et la nouvelle solution initiale générée. Le degré de la perturbation dans une ILS est un élément fondamental. En effet, si la perturbation est trop faible, la nouvelle solution de départ sera a priori dans la même région de recherche et la recherche locale risque de retomber sur le même optimum local. À l'inverse, une perturbation trop forte risque d'avoir le même effet qu'une stratégie complètement aléatoire de génération du nouveau point de départ, avec ses avantages et ses inconvénients tels que discuté dans le paragraphe précédent.

Pour terminer cette section, nous soulignons que d'autres mécanismes d'échappement existent dans la littérature notamment dans le cadre d'autres méta-heuristiques et algorithmes évolutionnaires tels que le recuit simulé [69, 51, 16], la recherche tabou [29], les algorithmes mémétiques [76], et bien d'autres.

Dans cette thèse, notre but est d'augmenter la puissance de MOEA/D pour l'optimisation multi-objectif par des mécanismes d'échappement. À cet effet, les mécanismes standard utilisés en optimisation mono-objectif dans le cadre d'une ILS se sont imposés par leur simplicité et leur adéquation par rapport au concept de décomposition de MOEA/D. Cependant, d'autres mécanismes d'échappement de la littérature peuvent

constituer autant d'alternatives possibles, que nous n'aborderons pas dans ce manuscrit.

3.1.3 Optima locaux en optimisation multi-objectif

La définition d'un optimum local se généralise facilement dans le contexte de l'optimisation multi-objectif en se basant sur le concept de dominance. On peut en effet dire qu'une solution est un optimum local de Pareto s'il n'existe aucune solution dans son voisinage qui la domine (selon la définition de la relation de dominance de Pareto) [79]. Il est tout aussi clair que pouvoir s'échapper des optima locaux de Pareto est un enjeu crucial pour mieux approcher le front Pareto.

D'un point de vue général, les algorithmes évolutionnaires multi-objectifs sont basés sur l'idée de faire évoluer une population de solutions en cherchant des solutions enfants améliorantes. La notion d'amélioration reste cependant plus floue qu'en optimisation mono-objectif, puisque chaque type d'algorithmes adopte en général un paradigme de recherche multi-objectif qui lui est propre. Autrement dit, le concept d'optimalité locale selon la relation de dominance n'est pas toujours le plus adéquat pour appréhender le fait que la population maintenue par un processus de recherche inhérent à un algorithme particulier est coincée dans des régions non-améliorantes.

On peut trouver un large éventail d'algorithmes multi-objectifs qui tentent d'adapter des principes issus des de l'optimisation mono-objectif pour éviter d'être piégé dans des zones d'optimum local de Pareto. À titre d'exemple, dans [106], les auteurs intègrent l'algorithme ILS pour un problème de flowshop de permutation multi-objectif. La recherche locale mono-objectif est remplacée par une recherche locale Pareto (*Pareto Local Search*) avec une profondeur de recherche qui est modifiée au fur et à mesure des itérations. Chaque recherche locale Pareto débute avec une solution non-dominée trouvée auparavant, qui est ensuite perturbée pour s'éloigner de la zone d'optimum locale de Pareto. Dans [28], Gandibleux et al. utilisent une approche par recherche tabou pour approximer le front Pareto. De la même façon, le recuit simulé est une méta-heuristique très étudiée qui s'est vue être étendue à de nombreuses reprises dans le cadre de problèmes d'optimisation combinatoire multi-objectif [2]. Pour transposer cet algorithme dans un contexte multi-objectif, les deux grandes techniques utilisées sont l'utilisation d'une population de solutions [20, 58] ou de relations de dominance [93, 4] pour guider la recherche vers le front Pareto tout en évitant les optima locaux de Pareto. D'autres techniques plus implicites ont également été proposées pour détecter directement les optima locaux de Pareto comme les travaux de Paquete et al. [78]. Les études cherchant à trouver le meilleur critère d'arrêt [99, 12] peuvent aussi être vues

comme des indicateurs de détection d'optima locaux en optimisation multi-objectif, où le but est d'arrêter automatiquement l'algorithme quand le processus de recherche est bloqué.

Dans le cadre des algorithmes multi-objectif à base de décomposition comme MOEA/D, le problème multi-objectif initial est décomposé en plusieurs sous-problèmes mono-objectifs. Il devient donc naturel de considérer que certaines solutions peuvent être des optima locaux pour un sous-problème scalaire donnée, selon la définition mono-objectif d'optimalité locale. Autrement dit, nous pouvons naturellement se baser sur la notion d'optima locaux scalaires [11], par rapport à chacun des sous problèmes mono-objectifs définis dans MOEA/D. Ces optima locaux scalaires sont les optima pouvant être atteints par les différents sous-problèmes définis par la fonction d'agrégation utilisée par MOEA/D. On considère alors qu'un sous-problème a atteint un optimum local scalaire lorsque le processus de recherche n'arrive plus à améliorer la solution liée à ce sous-problème. Cette remarque simple est à la base de la contribution présentée dans ce chapitre, et de façon indirecte à la base d'autres variantes de MOEA/D dans la littérature. En particulier, les techniques utilisées pour détecter les sous-problèmes difficiles [117, 82] comme dans MOEA/D–DRA par exemple (voir chapitre précédent), peuvent être vues comme des techniques se basant sur l'identification d'optima locaux scalaires. En effet, pour détecter la présence de sous-problèmes difficiles, MOEA/D–DRA cherche à trouver les sous-problèmes qui ne permettent plus d'avancer vers le front Pareto, et qui sont donc a priori coincés dans des optima locaux.

Dans la suite de ce chapitre, nous montrons qu'en s'inspirant de la littérature en optimisation mono-objectif, des mécanismes simples d'échappement peuvent être intégrés dans MOEA/D, nous permettant ainsi d'augmenter la puissance du framework pour la résolution de problème multi-objectif.

3.2 Intégration d'un mécanisme d'échappement dans MOEA/D

Dans MOEA/D, la décomposition des sous-problèmes permet de faire évoluer les solutions maintenues par les différents sous-problèmes de façon coopérative. Chaque solution liée à un sous-problème permet d'aider à l'amélioration des sous-problèmes voisins. Cependant, il se peut que l'optimisation de certains sous-problèmes soit bloquée dans certaines régions. Le fait qu'un sous-problème ne soit plus amélioré impacte évidemment ses sous-problèmes voisins, puisqu'ils ne peuvent plus profiter de ces

Algorithme 4 : Template de MOEA/D-Escape

Input : $\mathcal{W} := \left\{\omega^1, \ldots, \omega^\mu\right\}$: poids ; $g(\cdot \mid \omega)$: fonction d'agrégation ; T : taille du voisinage ; τ : seuil de détection ; H : taille de l'historique des amélioration ;

$\mathcal{EP} \leftarrow \emptyset$: (Facultatif) Population Externe ;
$\mathcal{P} \leftarrow \left\{x^1, \ldots, x^\mu\right\}$: génère et évalue la population initiale de taille μ;
$z^\star \leftarrow$ initialise le point de référence avec $\mathcal{P}$;
$g \leftarrow 0$ // initialisation du compteur de génération
tant que *le critère d'arrêt n'est pas atteint* **faire**
    $g \leftarrow g + 1$;
    pour $w^i \in \mathcal{W}$ **faire** $I_i^g \leftarrow 0$;
    pour $w^i \in \mathcal{W}$ **faire**
        $\mathcal{B}_i \leftarrow$ voisinage du sous-problème w^i avec les T plus proches sous-problème;
        $\mathcal{X} \leftarrow$ sélection des parents potentiels dans le voisinage $\mathcal{B}_i$;
        $x' \leftarrow$ utilisation d'opérateurs de variation pour générer une solution enfant à partir de $\mathcal{X}$;
        $F(x') \leftarrow$ évaluation de la solution enfant x' ;
        $z^\star \leftarrow$ mise à jour du point de référence avec $F(x')$;
        $\mathcal{EP} \leftarrow$ mise à jour de la population externe des solutions non-dominées avec la nouvelle solution x';
        // Remplacement de la solution x' dans la population
        pour $j \in \mathcal{B}_i$ **faire**
            si $g(x' \mid \omega^j) < g(x^j \mid \omega^j)$ **alors**
                $\mathcal{P}_\mu^j \leftarrow x'$;
                $I_i^g \leftarrow I_i^g + 1$;
        $\mathcal{P} \leftarrow$ processus de remplacement de la solution x' dans la population $\mathcal{P}$;
    // Calcul du signal d'activation
    $\mu_g^+ \leftarrow \sum_{i=1}^{\mu} I_i^g$;
    $p^+ \leftarrow \frac{1}{H} \cdot \sum_{\ell=\max\{1,g-H\}}^{g} \frac{\mu_\ell^+}{\mu}$;
    // Exécution de la stratégie d'échappement
    si $p^+ \leq \tau$ **alors**
        $\mathcal{P}_\mu \leftarrow$ perturbation de la population selon la stratégie choisit;

améliorations pour eux-mêmes, et ainsi permettre au processus de continuer à améliorer globalement la qualité de l'ensemble des solutions maintenues dans la population.

Lorsque MOEA/D est coincé dans certaines régions, l'idée générale que nous essayons de développer ici est de lui permettre de s'échapper des régions d'attraction actuelles, pour se réorienter vers des régions plus intéressantes et mieux s'approcher du front Pareto. Ceci nécessite de répondre à deux questions essentielles :

— Comment détecter que le processus de recherche inhérent à MOEA/D est bloqué ?
— Comment permettre à MOEA/D de s'échapper vers des régions de recherche plus bénéfiques ?

Afin de répondre à ces deux questions, nous proposons d'étendre le framework MOEA/D avec des composants de détection et d'échappement simples, tel que décrit dans le template de l'Algorithme 4. Il est à noter que les différences principales avec la version standard de MOEA/D sont montrées en utilisant un fond gris dans l'Algorithme 4, ceci afin de mieux apprécier la généricité et la simplicité de notre nouvelle proposition. La suite de cette section est dédiée à une description détaillée des solutions que nous proposons pour répondre aux deux questions précédemment énoncées.

3.2.1 Détection des optima locaux dans MOEA/D

Pour pouvoir détecter que la population dans MOEA/D est bloquée, nous nous appuyons sur le fait que MOEA/D parcours les sous-problèmes à la recherche d'améliorations. Nous allons ainsi pouvoir observer, à chaque génération du processus évolutionnaire, *le nombre d'améliorations* effectuées dans la population, lors du parcours des différents sous-problèmes. Le nombre d'améliorations est initialisé et mis à jour aux lignes 7 et 18 de l'Algorithme 4. On considère qu'une modification à lieu quand une solution nouvellement générée remplace une solution de la population $\mathcal{P}$.

L'indicateur μ_g^+ (ligne 21 de l'Algorithme 4) correspond au nombre d'améliorations réalisées au sein de la population pendant la génération g. Cet indicateur nous permet d'identifier la difficulté que rencontre le processus évolutionnaire pour améliorer la population. La valeur de cet indicateur est comprise entre 0 et $\mu \times T$. Si la valeur est égale à $\mu \times T$, cela signifie que chaque solution x' qui a été générée pendant la génération g, a pu améliorer les T solutions présentes dans le voisinage $\mathcal{B}_i$. Au contraire si la valeur de μ_g^+ est égale à 0, cela signifie qu'aucune des solutions générées pendant la génération g n'ont permis d'améliorer la population $\mathcal{P}$.

En conséquence, le ratio μ_g^+/μ peut être considéré comme un indicateur qui représente la probabilité que le processus de recherche soit bloqué pendant la génération g.

D'une manière générale, cette probabilité (empirique) μ_g^+/μ peut dépendre de plusieurs facteurs. Le premier facteur important influant sur le nombre d'améliorations réalisées provient de la qualité des solutions qui ont pu être générés par les opérateurs de variation. En effet, ces opérateurs sont stochastiques et peuvent dépendre des solutions utilisées comme parents pour générer les solutions enfants. La probabilité d'améliorer la population dépend également de l'avancée du processus de recherche. En effet, on peut raisonnablement s'attendre à voir cette probabilité décroître au fur et à mesure de l'avancement du processus de recherche, puisqu'il est a priori plus difficile de trouver des solutions améliorantes au fur et à mesure que l'on se rapproche du front Pareto.

Dans notre contexte d'étude, il est également raisonnable d'émettre l'hypothèse que cette probabilité peut : (1) décroître quand des sous-problèmes se retrouveront bloqués dans certaines régions de l'espace de recherche, et (2) croître de nouveau et de façon ponctuelle lorsque le processus de recherche n'est plus bloqué. En effet, il se peut que certains sous-problèmes qui étaient bloqués depuis un certain temps soit débloqués par des sous-problèmes voisins. De ce fait, il faut pouvoir traquer l'évolution de l'indicateur μ_g^+/μ non pas sur la dernière génération exécutée, mais sur les H dernières générations. Cette valeur est ainsi stockée dans la variable p^+ (ligne 22 de l'Algorithme 4) nous permettant ainsi de concevoir un signal d'activation pour notre mécanisme d'échappement. Autrement dit, quand la valeur de p^+ passe sous un seuil τ défini en paramètre, l'algorithme considère avoir détecté que la recherche est bloquée et pourra ainsi activer le système d'échappement (ligne 23 de l'Algorithme 4).

3.2.2 Échappement aux optima locaux dans MOEA/D

Une fois que le signal d'activation est déclenché, c'est-à-dire lorsque $p^+ < \tau$ à la ligne 23 de l'Algorithme 4, le système d'échappement est exécuté. Le but de ce composant est de s'éloigner des zones qui bloquent actuellement le processus de recherche. Dans cette thèse, nous étudions les deux méthodes d'échappement inspirées par les recherches locales itérées en optimisation mono-objectif, telles que discutées précédemment en introduction de ce chapitre :

— *Échappement par réinitialisation aléatoire de la population.* La première stratégie d'échappement consiste à réinitialiser la population $\mathcal{P}$ par une nouvelle population constituée de solutions générées aléatoirement. Cette stratégie, notée $\mathsf{Escape}_{\mathsf{Random}}$, est montré en pseudo-code dans l'Algorithme 5. On peut noter que seule la population $\mathcal{P}$ est réinitialisée, la population externe $\mathcal{EP}$ est conservée pour continuer de recevoir les nouvelles solutions non-dominées qui seront

Algorithme 5 : Stratégie d'échappement aléatoire $\text{Escape}_{\text{Random}}$

```
1 R
  Input : P : population de taille μ ;
2 pour x^i ∈ P faire
      // Remplacement de la solution x^i de la population par une
         nouvelle solution aléatoire
3     x^i ← nouvelle solution générée aléatoirement
```

Algorithme 6 : Stratégie d'échappement par mutation $\text{Escape}_{\text{Mutation}}$

```
  Input : P : population de taille μ ; c : taux de mutation (c < n)
1 pour x^i ∈ P faire
      // Parcours de chaque bits j de la solution x^i
2     pour j ∈ {0,...,n} faire
3         si (c/n) < nombre aléatoire entre 0 et 1 alors
4             x^i_j ← ¬(x^i_j)
```

trouvées à partir de la nouvelle population de départ.

— *Échappement par mutation au sein de la population.* La seconde stratégie d'échappement consiste à effectuer une mutation stochastique sur *chacune* des solutions constituant la population $\mathcal{P}$. En pratique, pour les problèmes binaires que nous considérons dans notre étude expérimentale, ceci se traduit par le fait de muter chaque bit d'une solution avec une probabilité c/n, où n est la taille du problème et c un paramètre permettant de fixer le degré de la perturbation. Cette stratégie, notée $\text{Escape}_{\text{Mutation}}$, est disponible en pseudo-code dans l'Algorithme 6. Ce type de perturbation nous permettra par la suite d'étudier l'impact du degré de perturbation sur le processus de recherche multi-objectif, ce qui est en soi une question importante.

3.3 Analyse expérimentale

La dernière section de ce chapitre s'intéresse à l'analyse expérimentale de notre mécanisme d'échappement. Nous commençons d'abord par décrire notre protocole expérimental. Ensuite, nous présentons l'étude du mécanisme de détection ainsi que celui de perturbation, en se basant à la fois sur la version de base de MOEA/D et sa variante MOEA/D-(μ, λ, *sps*) décrite dans le chapitre précédent.

3.3.1 Protocole

3.3.1.1 Problèmes étudiés

De même que dans le chapitre précédent, nous considérons des instances du problème mnk-landscapes avec des niveaux de difficulté différents. Plus précisément, nous considérons des instances avec 2 objectifs ($m = 2$), et des dimensions $n \in \{50, 100, 200\}$. Les degrés de rugosité des instances sont variables, en considérant des valeurs d'épistasie $k \in \{1, 4\}$. Rappelons ici qu'augmenter le degré d'épistasie k a pour impact connu d'augmenter le nombre d'optima locaux. Les instances avec $k = 1$ sont relativement lisses alors que les instances avec $k = 4$ sont très rugueuses. Grâce à ces 12 instances, nous considérons ainsi un panel raisonnablement large nous permettant d'analyser correctement nos stratégies d'échappement dans MOEA/D.

3.3.1.2 Algorithmes étudiés et paramètres

Dans ce qui suit, nous allons essentiellement comparer les performances de MOEA/D avec et sans mécanisme d'échappement. Comme expliqué précédemment, nous considérons deux stratégies de perturbation. La stratégie de réinitialisation aléatoire de la population (Algorithme 5) est notée $\mathsf{Escape}_{\mathsf{Random}}$. La seconde stratégie d'échappement qui utilise un opérateur de mutation (Algorithme 6) est notée $\mathsf{Escape}_{\mathsf{Mutation}}$. La stratégie $\mathsf{Escape}_{\mathsf{Mutation}}$ prend comme paramètre la probabilité de mutation c. Nous étudions les valeurs suivantes du degré de perturbation $c \in \{0.1 \cdot n, 0.2 \cdot n, 0.3 \cdot n\}$. Avec ces trois valeurs, on considère alors que l'on perturbera respectivement en moyenne 10%, 20% et 30% de chaque solution de la population. Pour plus de simplicité, on utilisera alors les notations $\mathsf{Escape}_{\mathsf{Mutation}}10\%$, $\mathsf{Escape}_{\mathsf{Mutation}}20\%$ et $\mathsf{Escape}_{\mathsf{Mutation}}30\%$, pour faire référence aux différentes valeurs du degré de perturbation.

Les paramètres de MOEA/D communs à chaque configuration sont ceux utilisés couramment dans la littérature. Ainsi, la taille de population est fixée à 500 solutions. La taille du voisinage T est fixée à 20% de la taille de la population. Les 500 vecteurs de poids utilisés pour décomposer le problème multi-objectif sont générés en suivant la méthodologie SOBOL [115]. Pour générer les nouvelles solutions pendant le processus évolutionnaire de reproduction, les solutions parents se composent de la solution associée au sous-problème actuellement parcouru et d'une solution choisie aléatoirement dans le voisinage du sous-problème courant. Une nouvelle solution enfant est générée en utilisant un opérateur de croisement à 2 points, suivi d'un opérateur de mutation qui modifie chaque variable booléenne avec un taux de $1/n$.

Le mécanisme que nous proposons pour détecter le blocage du processus de recherche utilise deux paramètres. La fenêtre glissante de taille H sauvegarde les valeurs de notre indicateur μ_g^+ pour les H dernières générations. Le paramètre H nous permet de déterminer s'il vaut mieux traquer les blocages sur le long ou sur le court terme. Les différentes valeurs qui sont étudiées sont $H \in \{1, 200, 500, 1\,000, 2\,000, 5\,000\}$. On peut noter qu'une fenêtre de taille $H = 2\,000$ signifie que l'on traque le nombre d'améliorations sur les 10^6 dernières évaluations. Le second paramètre utilisé pour détecter le moment où il faut perturber la population est le seuil de détection τ. Les différentes valeurs considérées sont $\tau \in \{0, 0.01, 0.1\}$.

Les configurations possibles de chacun de ces paramètres seront étudiées pour la version originale de MOEA/D, que l'on peut aussi noter MOEA/D-($\mu = 500$, $\lambda = \mu$, sps_{ALL}) en adoptant les notations du chapitre précédent. On étudiera également chacune des configurations avec l'algorithme MOEA/D-($\mu = 500$, $\lambda = 1$, sps_{RND}) qui a globalement obtenu les meilleurs résultats dans le chapitre précédent. Le fait d'étudier le mécanisme d'échappement à la fois pour MOEA/D et MOEA/D-(μ, λ, sps) permet d'analyser la compatibilité entre les deux composants proposés dans cette thèse : le composant de répartition de l'effort de calcul dans MOEA/D, et celui d'échappement étudié spécifiquement dans ce chapitre.

3.3.1.3 Évaluation des performances

En considérant chaque combinaison de paramètres à étudier pour chaque instance de mnk-landscapes, il existe 432 configurations différentes. Pour réaliser nos expérimentations dans un temps raisonnable, nous exécutons chaque configuration dix fois indépendamment. Pour visualiser les profils de convergence de chaque configuration, nous considérons plusieurs conditions d'arrêt pour un budget maximal de 10^7 évaluations ($10^0, 10^1, ..., 10^7$).

L'indicateur de qualité utilisé pour présenter la qualité des solutions obtenues sera la déviation relative d'hypervolume décrit dans la Section 1.5.1.1 du Chapitre 1. Nous utilisons cet indicateur calculé par rapport à la population externe (l'archive) qui stocke toutes les solutions non-dominées trouvées par l'algorithme, que cela soit avant ou après les perturbations réalisées au sein de la population.

3.3.2 Étude des paramètres de détection

Dans cette section, nous étudions les deux paramètres pouvant impacter le système de détection, c'est-à-dire le seuil de détection τ et la taille de la fenêtre H qui stocke les

valeurs des indicateurs d'amélioration des sous-problèmes. Cette étude est effectuée sur l'algorithme MOEA/D-($\mu = 500$, $\lambda = \mu$, sps_{ALL}).

3.3.2.1 Étude des performances générales

Nous nous intéressons dans un premier temps aux performances générales des paramètres de détection après 10^7 évaluations pour avoir une vue globale sur l'impact de ces paramètres. Pour rappel, le seuil de détection τ permet de définir à l'aide du signal d'activation qui est calculé à chaque génération, le moment où il faut exécuter la stratégie d'échappement. La fenêtre glissante, de taille H, stocke quant à elle les informations sur l'amélioration de la population.

Pour étudier les performances générales de ces deux paramètres, nous utiliserons les Tableaux 3.1, 3.2, 3.3 et 3.4. Ces tableaux montrent les rangs de chacune des configurations possibles avec un budget de 10^7 évaluations pour les différentes valeurs de H et τ étudiées dans ce chapitre. Les différentes configurations des paramètres sont confrontées les unes avec les autres pour une stratégie d'échappement donnée et pour toutes les instances de mnk-landscapes étudiées. Les rangs de la stratégie d'échappement $\text{Escape}_{\text{Random}}$ sont étudiés dans le Tableau 3.1, alors que ceux pour les stratégies d'échappement $\text{Escape}_{\text{Mutation}}$ sont étudiés dans les Tableaux 3.2, 3.3 et 3.4 pour des taux de mutation de respectivement 10%, 20% et 30%. Pour rappel, les rangs représentent le nombre de configurations qui ont surpassées significativement la configuration étudiée en utilisant la valeur de la déviation relative de l'hypervolume selon un test de Wilcoxon. On cherche donc pour chaque configuration à avoir le rang le plus faible.

Étude de la stratégie $\text{Escape}_{\text{Random}}$ après 10^7 évaluations (Tableau 3.1). Tout d'abord, nous remarquons dans le Tableau 3.1 que les meilleures performances de la stratégie $\text{Escape}_{\text{Random}}$ sont obtenues avec un seuil de détection $\tau = 0$ et cela, quelle que soit la taille de l'instance ou son degré de rugosité. On peut notamment remarquer qu'en augmentant la valeur du seuil, on dégrade fortement les performances en obtenant les pires résultats toutes stratégies confondues quand nous choisissons le plus grand seuil de détection étudié dans ce chapitre $\tau = 0.1$.

Pour affiner notre analyse, on peut maintenant s'intéresser à la taille de la fenêtre H qui, même si elle semble avoir un impact moins important que le seuil de détection sur la stratégie $\text{Escape}_{\text{Random}}$, joue tout de même un rôle dans les performances de la stratégie de perturbation. On peut voir que pour un seuil de détection $\tau \leq 0.01$, il est préférable de stocker l'historique des améliorations sur un nombre de générations très important. En effet, plus la taille de la fenêtre H est importante et meilleurs sont les

Tableau 3.1 – Rangs et moyenne (entre parenthèse) de la déviation relative d'hypervolume, obtenu après 10^7 évaluations pour la stratégie d'échappement $\mathsf{Escape}_{\mathsf{Random}}$.Les valeurs d'hypervolume soulignées correspondent à la meilleure valeur moyenne.

			$k=1$			$k=4$		
			$n=50$	$n=100$	$n=200$	$n=50$	$n=100$	$n=200$
$\mathsf{Escape}_{\mathsf{Random}}$	$\tau=0$	$H=1$	$10_{(0.649)}$	$10_{(6.829)}$	$10_{(23.635)}$	$10_{(4.213)}$	$10_{(24.352)}$	$10_{(44.545)}$
		$H=200$	$0_{(\underline{0.1})}$	$0_{(0.207)}$	$4_{(1.406)}$	$0_{(\underline{0.104})}$	$4_{(3.312)}$	$4_{(11.852)}$
		$H=500$	$0_{(\underline{0.1})}$	$0_{(\underline{0.18})}$	$3_{(1.092)}$	$0_{(0.107)}$	$0_{(2.625)}$	$2_{(9.999)}$
		$H=1\,000$	$0_{(\underline{0.1})}$	$0_{(0.199)}$	$0_{(0.826)}$	$0_{(0.109)}$	$0_{(2.404)}$	$0_{(8.905)}$
		$H=2\,000$	$0_{(\underline{0.1})}$	$0_{(0.193)}$	$0_{(0.787)}$	$0_{(0.109)}$	$0_{(\underline{2.124})}$	$0_{(8.713)}$
		$H=5\,000$	$0_{(\underline{0.1})}$	$0_{(0.203)}$	$0_{(\underline{0.615})}$	$1_{(0.178)}$	$0_{(2.224)}$	$0_{(\underline{8.256})}$
	$\tau=0.01$	$H=1$	$10_{(0.599)}$	$11_{(7.917)}$	$11_{(25.874)}$	$10_{(4.781)}$	$11_{(25.863)}$	$11_{(46.52)}$
		$H=200$	$8_{(0.301)}$	$8_{(5.292)}$	$8_{(17.794)}$	$8_{(1.66)}$	$8_{(19.171)}$	$8_{(38.202)}$
		$H=500$	$8_{(0.344)}$	$8_{(5.286)}$	$8_{(17.246)}$	$8_{(1.461)}$	$8_{(18.915)}$	$8_{(38.361)}$
		$H=1\,000$	$7_{(0.173)}$	$7_{(4.244)}$	$7_{(16.511)}$	$7_{(0.867)}$	$7_{(16.975)}$	$7_{(36.331)}$
		$H=2\,000$	$0_{(0.11)}$	$6_{(2.614)}$	$6_{(14.347)}$	$6_{(0.535)}$	$6_{(14.542)}$	$6_{(34.59)}$
		$H=5\,000$	$0_{(\underline{0.1})}$	$5_{(0.746)}$	$5_{(9.222)}$	$5_{(0.232)}$	$5_{(9.776)}$	$5_{(27.957)}$
	$\tau=0.1$	$H=1$	$12_{(1.638)}$	$12_{(16.309)}$	$12_{(38.651)}$	$12_{(7.639)}$	$12_{(31.099)}$	$12_{(53.312)}$
		$H=200$	$13_{(1.966)}$	$13_{(23.98)}$	$13_{(52.101)}$	$12_{(7.348)}$	$13_{(35.555)}$	$13_{(60.896)}$
		$H=500$	$14_{(2.744)}$	$14_{(25.348)}$	$14_{(53.138)}$	$14_{(9.672)}$	$14_{(37.069)}$	$14_{(62.318)}$
		$H=1\,000$	$14_{(2.744)}$	$14_{(25.348)}$	$14_{(53.138)}$	$14_{(9.672)}$	$14_{(37.069)}$	$14_{(62.318)}$
		$H=2\,000$	$14_{(2.744)}$	$14_{(25.348)}$	$14_{(53.138)}$	$14_{(9.672)}$	$14_{(37.069)}$	$14_{(62.318)}$
		$H=5\,000$	$14_{(2.744)}$	$14_{(25.348)}$	$14_{(53.138)}$	$14_{(9.672)}$	$14_{(37.069)}$	$14_{(62.318)}$

résultats obtenus. On peut toutefois remarquer que ce comportement est inversé avec un seuil de détection inadéquat. En effet, quand le seuil est fixé $\tau = 0.1$, alors les meilleurs résultats sont obtenus avec les plus petites valeurs de H.

Étude de la stratégie $\mathsf{Escape}_{\mathsf{Mutation}}$ 10% après 10^7 évaluations (Tableau 3.2). La stratégie $\mathsf{Escape}_{\mathsf{Mutation}}$ 10% donne de très bons résultats sur l'instance $n = 50$ et $k = 1$, où toutes les configurations étudiées obtiennent de très bonnes performances avec un rang moyen à 0. Cette stratégie obtient également les meilleurs résultats pour l'instance $n = 50$ et $k = 4$, sauf lorsque le seuil le plus faible $\tau = 0$ est utilisé.

Mis à part ces observations sur les instances de dimension $n = 50$, les meilleurs résultats de cette stratégie sont obtenus lorsque le seuil de détection est inférieur ou égal à 0.01. Contrairement à la stratégie précédente, où les meilleurs résultats étaient obtenus avec un seuil $\tau = 0$ uniquement, les résultats de la stratégie $\mathsf{Escape}_{\mathsf{Mutation}}$ 10% sont ici légèrement différents. En effet, pour les instances avec un degré d'épistasie $k = 1$, les meilleurs résultats sont obtenus avec un seuil $\tau = 0$, alors que pour les instances plus rugueuses avec $k = 4$, le seuil $\tau = 0.01$ s'avère préférable. La stratégie $\mathsf{Escape}_{\mathsf{Mutation}}$

Tableau 3.2 – Rangs et moyenne (entre parenthèse) de la déviation relative d'hypervolume, obtenu après 10^7 évaluations pour la stratégie d'échappement $\textsf{Escape}_{\textsf{Mutation}}$ 10%.Les valeurs d'hypervolume soulignées correspondent à la meilleure valeur moyenne.

			$k=1$			$k=4$		
			$n=50$	$n=100$	$n=200$	$n=50$	$n=100$	$n=200$
$\textsf{Escape}_{\textsf{Mutation}}$ 10%	$\tau=0$	$H=1$	$0_{(\underline{0.1})}$	$9_{(2.018)}$	$10_{(11.858)}$	$0_{(0.543)}$	$10_{(7.253)}$	$10_{(30.647)}$
		$H=200$	$0_{(\underline{0.1})}$	$3_{(0.827)}$	$3_{(2.253)}$	$13_{(2.808)}$	$0_{(2.907)}$	$0_{(6.605)}$
		$H=500$	$0_{(\underline{0.1})}$	$2_{(0.79)}$	$1_{(1.876)}$	$13_{(2.809)}$	$0_{(3.29)}$	$0_{(\underline{5.756})}$
		$H=1\,000$	$0_{(\underline{0.1})}$	$1_{(0.744)}$	$0_{(1.96)}$	$13_{(3.006)}$	$0_{(3.913)}$	$0_{(6.806)}$
		$H=2\,000$	$0_{(\underline{0.1})}$	$1_{(0.738)}$	$0_{(\underline{1.464})}$	$13_{(3.775)}$	$0_{(4.326)}$	$0_{(6.211)}$
		$H=5\,000$	$0_{(\underline{0.1})}$	$0_{(\underline{0.655})}$	$0_{(1.808)}$	$13_{(3.924)}$	$0_{(4.433)}$	$0_{(6.651)}$
	$\tau=0.01$	$H=1$	$0_{(0.11)}$	$9_{(2.277)}$	$11_{(13.008)}$	$0_{(0.557)}$	$11_{(8.083)}$	$11_{(32.45)}$
		$H=200$	$0_{(\underline{0.1})}$	$10_{(2.285)}$	$9_{(11.112)}$	$0_{(0.673)}$	$7_{(5.883)}$	$9_{(22.475)}$
		$H=500$	$0_{(\underline{0.1})}$	$8_{(1.653)}$	$8_{(10.45)}$	$7_{(0.825)}$	$5_{(4.572)}$	$8_{(20.838)}$
		$H=1\,000$	$0_{(\underline{0.1})}$	$7_{(1.413)}$	$7_{(9.276)}$	$0_{(0.703)}$	$0_{(3.047)}$	$7_{(17.876)}$
		$H=2\,000$	$0_{(\underline{0.1})}$	$6_{(1.293)}$	$6_{(7.637)}$	$0_{(0.609)}$	$1_{(3.092)}$	$5_{(15.179)}$
		$H=5\,000$	$0_{(\underline{0.1})}$	$5_{(1.046)}$	$5_{(5.609)}$	$0_{(\underline{0.46})}$	$0_{(\underline{2.598})}$	$5_{(14.783)}$
	$\tau=0.1$	$H=1$	$0_{(0.112)}$	$12_{(3.419)}$	$12_{(18.4)}$	$0_{(0.547)}$	$12_{(9.703)}$	$12_{(37.688)}$
		$H=200$	$0_{(\underline{0.1})}$	$13_{(3.722)}$	$13_{(20.145)}$	$0_{(0.52)}$	$12_{(10.189)}$	$13_{(40.588)}$
		$H=500$	$0_{(\underline{0.1})}$	$13_{(3.722)}$	$13_{(20.145)}$	$0_{(0.52)}$	$12_{(10.189)}$	$13_{(40.588)}$
		$H=1\,000$	$0_{(\underline{0.1})}$	$13_{(3.722)}$	$13_{(20.145)}$	$0_{(0.52)}$	$12_{(10.189)}$	$13_{(40.588)}$
		$H=2\,000$	$0_{(\underline{0.1})}$	$13_{(3.722)}$	$13_{(20.145)}$	$0_{(0.52)}$	$12_{(10.189)}$	$13_{(40.588)}$
		$H=5\,000$	$0_{(\underline{0.1})}$	$13_{(3.722)}$	$13_{(20.145)}$	$0_{(0.52)}$	$12_{(10.189)}$	$13_{(40.588)}$

10% étant celle qui perturbe le moins la population, on peut donc penser que le seuil de détection doit être moins strict pour les instances avec de nombreux optima locaux ($k=4$), ceci afin de déclencher le mécanisme d'échappement plus rapidement. On remarque également que pour les instances de grande taille ($n=100$ et $n=200$), les moins bons résultats sont obtenus avec le plus grand seuil de détection, comme pour la stratégie $\textsf{Escape}_{\textsf{Random}}$. De même, le comportement du paramètre H semble être le même que pour la stratégie $\textsf{Escape}_{\textsf{Random}}$. En effet, on peut remarquer que pour les seuils avec les meilleurs résultats ($\tau \leq 0.01$), plus la taille H de la fenêtre est importante et meilleurs sont les résultats obtenus.

Étude des stratégies $\textsf{Escape}_{\textsf{Mutation}}$ 20% et 30% après 10^7 évaluations. Étant donné que les comportements des stratégies $\textsf{Escape}_{\textsf{Mutation}}$ 20% et $\textsf{Escape}_{\textsf{Mutation}}$ 30% que l'on peut distinguer dans les Tableaux 3.3 et 3.4 semblent identiques, nous choisissons de les étudier ensemble pour plus de clarté.

Comme pour les stratégies précédentes, on remarque que le seuil de détection

Tableau 3.3 – Rangs et moyenne (entre parenthèse) de la déviation relative d'hypervolume, obtenu après 10^7 évaluations pour la stratégie $\textsf{Escape}_{\textsf{Mutation}}$ 20%.Les valeurs d'hypervolume soulignées correspondent à la meilleure valeur moyenne.

			$k=1$			$k=4$		
			$n=50$	$n=100$	$n=200$	$n=50$	$n=100$	$n=200$
$\textsf{Escape}_{\textsf{Mutation}}$ 20%	$\tau=0$	$H=1$	$8_{(0.125)}$	$10_{(3.852)}$	$10_{(16.567)}$	$10_{(0.9)}$	$10_{(17.583)}$	$10_{(40.004)}$
		$H=200$	$0_{(\underline{0.1})}$	$1_{(0.568)}$	$4_{(1.669)}$	$0_{(\underline{0.138})}$	$1_{(1.363)}$	$0_{(4.526)}$
		$H=500$	$0_{(\underline{0.1})}$	$0_{(0.533)}$	$1_{(1.33)}$	$1_{(0.286)}$	$0_{(0.944)}$	$0_{(\underline{3.375})}$
		$H=1\,000$	$0_{(\underline{0.1})}$	$0_{(0.546)}$	$1_{(1.315)}$	$3_{(0.363)}$	$0_{(1.303)}$	$0_{(4.078)}$
		$H=2\,000$	$0_{(\underline{0.1})}$	$0_{(0.51)}$	$0_{(1.042)}$	$1_{(0.247)}$	$0_{(\underline{0.875})}$	$0_{(4.037)}$
		$H=5\,000$	$0_{(\underline{0.1})}$	$0_{(\underline{0.476})}$	$0_{(\underline{0.827})}$	$1_{(0.405)}$	$0_{(1.572)}$	$0_{(4.699)}$
	$\tau=0.01$	$H=1$	$8_{(0.165)}$	$11_{(4.501)}$	$11_{(18.663)}$	$10_{(0.89)}$	$10_{(18.63)}$	$10_{(41.155)}$
		$H=200$	$8_{(0.18)}$	$9_{(3.586)}$	$8_{(13.328)}$	$5_{(0.441)}$	$9_{(13.358)}$	$8_{(32.698)}$
		$H=500$	$0_{(0.122)}$	$8_{(3.124)}$	$8_{(12.93)}$	$2_{(0.42)}$	$8_{(11.697)}$	$8_{(31.589)}$
		$H=1\,000$	$0_{(\underline{0.1})}$	$7_{(2.254)}$	$7_{(11.478)}$	$1_{(0.24)}$	$7_{(10.014)}$	$7_{(30.048)}$
		$H=2\,000$	$0_{(\underline{0.1})}$	$6_{(1.356)}$	$6_{(9.243)}$	$0_{(0.322)}$	$6_{(7.413)}$	$6_{(26.24)}$
		$H=5\,000$	$0_{(\underline{0.1})}$	$5_{(0.794)}$	$5_{(6.207)}$	$0_{(0.268)}$	$5_{(6.067)}$	$5_{(22.675)}$
	$\tau=0.1$	$H=1$	$11_{(0.233)}$	$12_{(8.687)}$	$12_{(28.76)}$	$10_{(1.007)}$	$12_{(23.074)}$	$12_{(48.794)}$
		$H=200$	$12_{(0.326)}$	$13_{(10.515)}$	$13_{(34.975)}$	$10_{(1.007)}$	$13_{(25.431)}$	$13_{(54.781)}$
		$H=500$	$12_{(0.326)}$	$13_{(10.515)}$	$13_{(34.975)}$	$10_{(1.007)}$	$13_{(25.431)}$	$13_{(54.781)}$
		$H=1\,000$	$12_{(0.326)}$	$13_{(10.515)}$	$13_{(34.975)}$	$10_{(1.007)}$	$13_{(25.431)}$	$13_{(54.781)}$
		$H=2\,000$	$12_{(0.326)}$	$13_{(10.515)}$	$13_{(34.975)}$	$10_{(1.007)}$	$13_{(25.431)}$	$13_{(54.781)}$
		$H=5\,000$	$12_{(0.326)}$	$13_{(10.515)}$	$13_{(34.975)}$	$10_{(1.007)}$	$13_{(25.431)}$	$13_{(54.781)}$

le plus élevé ($\tau = 0.1$) ne permet pas d'atteindre de bons résultats pour aucune des instances étudiées. Comme pour la stratégie $\textsf{Escape}_{\textsf{Random}}$, les deux stratégies étudiées ici obtiennent les meilleurs résultats toutes instances confondues lorsque le seuil de détection le plus strict est utilisé ($\tau = 0$), c'est-à-dire quand on ne détecte aucune amélioration dans la fenêtre glissante pendant les dernières H générations.

En étudiant maintenant la taille H de la fenêtre glissante, on remarque que le comportement est identique aux stratégies précédemment étudiées et qu'il faut alors utiliser une grande taille d'historique. On peut toutefois noter qu'à partir d'un certain point, il ne semble plus nécessaire d'augmenter cette valeur quand le seuil est correctement paramétré ($\tau = 0$). On peut par exemple voir ce phénomène pour la stratégie $\textsf{Escape}_{\textsf{Mutation}}$ 20% sur les instances $n = 100$, ou pour la stratégie $\textsf{Escape}_{\textsf{Mutation}}$ 30% sur les instances $n = 200$.

3.3.2.2 Étude des profils de convergence

Intéressons-nous maintenant au profil de convergence du système de détection des optima locaux. La section précédente nous a permis de montrer les tendances générales

Tableau 3.4 – Rangs et moyenne (entre parenthèse) de la déviation relative d'hypervolume, obtenu après 10^7 évaluations pour la stratégie $\textsf{Escape}_{\textsf{Mutation}}$ 30%.Les valeurs d'hypervolume soulignées correspondent à la meilleure valeur moyenne.

			$k=1$			$k=4$		
			$n=50$	$n=100$	$n=200$	$n=50$	$n=100$	$n=200$
$\textsf{Escape}_{\textsf{Mutation}}$ 30%	$\tau=0$	$H=1$	$10_{(0.256)}$	$9_{(4.967)}$	$10_{(19.768)}$	$10_{(2.129)}$	$10_{(20.812)}$	$10_{(42.805)}$
		$H=200$	$0_{(\underline{0.1})}$	$1_{(0.369)}$	$3_{(1.273)}$	$0_{(0.131)}$	$2_{(2.004)}$	$3_{(8.537)}$
		$H=500$	$0_{(\underline{0.1})}$	$0_{(0.365)}$	$2_{(1.125)}$	$0_{(\underline{0.122})}$	$0_{(1.488)}$	$3_{(7.903)}$
		$H=1\,000$	$0_{(\underline{0.1})}$	$1_{(0.392)}$	$0_{(0.872)}$	$0_{(0.131)}$	$0_{(1.402)}$	$0_{(6.383)}$
		$H=2\,000$	$0_{(\underline{0.1})}$	$0_{(\underline{0.315})}$	$0_{(0.833)}$	$1_{(0.15)}$	$0_{(1.464)}$	$0_{(6.238)}$
		$H=5\,000$	$0_{(\underline{0.1})}$	$1_{(0.395)}$	$0_{(\underline{0.705})}$	$3_{(0.215)}$	$0_{(\underline{1.109})}$	$0_{(\underline{5.291})}$
	$\tau=0.01$	$H=1$	$10_{(0.327)}$	$11_{(5.875)}$	$11_{(22.17)}$	$10_{(2.476)}$	$11_{(22.714)}$	$10_{(44.09)}$
		$H=200$	$8_{(0.167)}$	$9_{(4.489)}$	$9_{(15.338)}$	$8_{(0.861)}$	$8_{(17.116)}$	$9_{(36.618)}$
		$H=500$	$8_{(0.192)}$	$8_{(3.941)}$	$8_{(14.826)}$	$8_{(0.864)}$	$8_{(16.136)}$	$8_{(35.524)}$
		$H=1\,000$	$0_{(\underline{0.1})}$	$7_{(3.117)}$	$7_{(13.82)}$	$5_{(0.432)}$	$7_{(14.143)}$	$7_{(34.15)}$
		$H=2\,000$	$0_{(\underline{0.1})}$	$6_{(1.541)}$	$6_{(11.095)}$	$5_{(0.524)}$	$6_{(10.392)}$	$6_{(31.252)}$
		$H=5\,000$	$0_{(\underline{0.1})}$	$5_{(0.793)}$	$5_{(7.087)}$	$5_{(0.442)}$	$5_{(8.409)}$	$5_{(25.711)}$
	$\tau=0.1$	$H=1$	$12_{(0.626)}$	$12_{(12.233)}$	$12_{(33.992)}$	$12_{(3.879)}$	$12_{(27.078)}$	$12_{(51.974)}$
		$H=200$	$12_{(0.934)}$	$13_{(16.971)}$	$13_{(44.181)}$	$12_{(4.527)}$	$13_{(32.412)}$	$13_{(59.172)}$
		$H=500$	$12_{(0.934)}$	$13_{(16.971)}$	$13_{(44.181)}$	$12_{(4.527)}$	$13_{(32.412)}$	$13_{(59.172)}$
		$H=1\,000$	$12_{(0.934)}$	$13_{(16.971)}$	$13_{(44.181)}$	$12_{(4.527)}$	$13_{(32.412)}$	$13_{(59.172)}$
		$H=2\,000$	$12_{(0.934)}$	$13_{(16.971)}$	$13_{(44.181)}$	$12_{(4.527)}$	$13_{(32.412)}$	$13_{(59.172)}$
		$H=5\,000$	$12_{(0.934)}$	$13_{(16.971)}$	$13_{(44.181)}$	$12_{(4.527)}$	$13_{(32.412)}$	$13_{(59.172)}$

des différents paramètres de détection des optima locaux après 10^7 évaluations. Nous allons maintenant pouvoir confronter ces observations avec les profils de convergence pour étudier le comportement des différents paramètres au cours du processus de recherche. Pour ce faire, nous étudierons la Figure 3.2 illustrant le profil de convergence des différentes stratégies en fonction des seuils de détection $\tau \in \{0, 0.01, 0.1\}$. Nous étudierons également le profil de convergence de ces mêmes stratégies en fonction des différentes tailles de fenêtres H avec les Figures 3.3 et 3.4 pour des seuils de détection de respectivement $\tau = 0$ et $\tau = 0.01$.

Étude du seuil de détection τ. Nous avons vu précédemment qu'il est préférable de choisir un seuil de détection $\tau = 0$ dans la plupart des cas quand nous avons un budget de 10^7 évaluations. Avec la Figure 3.2, nous pouvons voir que cette observation n'est pas systématique pour tous les budgets. En effet, on peut voir que la différence entre $\tau = 0$ et $\tau = 0.01$ n'est pas significative quand le budget est très faible. On peut voir sur la figure que les valeurs $\tau \geq 0.01$ surpassent assez rapidement la valeur $\tau = 0.1$ ($\leq 10^3$ évaluations) mais se démarquent l'une de l'autre qu'après 10^5 évaluations quand la

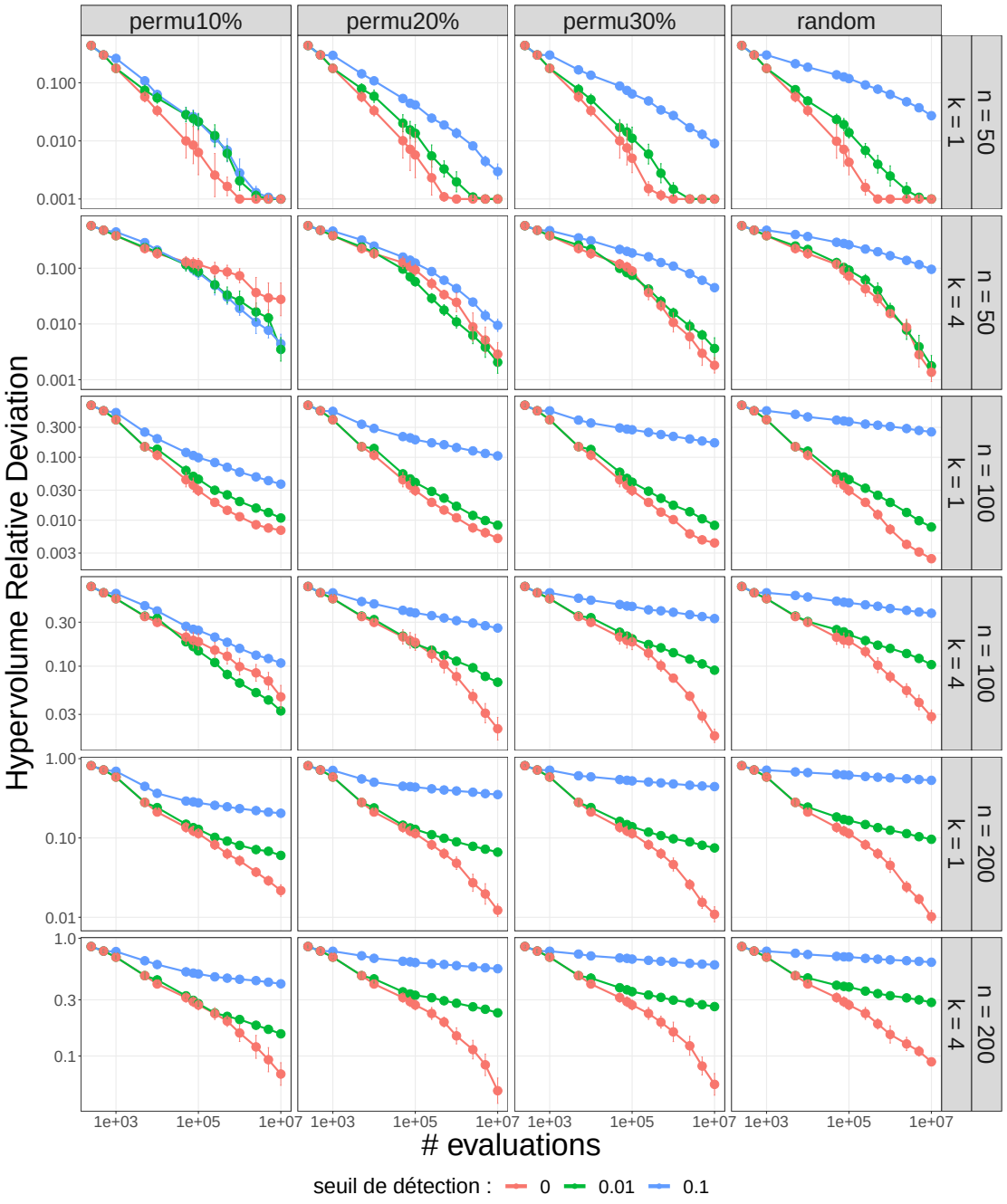

FIGURE 3.2 – Profils de convergence des différentes stratégies d'échappement avec une taille de fenêtre $H = 5\,000$.

valeur $\tau = 0.01$ ne semble plus suffire pour la majorité des instances.

On peut tout de même remarquer pour certaines instances spécifiques, comme avec l'instance où $n = 50$ et $k = 4$, que le seuil de détection semble plus difficile à paramétrer quand le nombre d'optima locaux est important. On remarque notamment pour les stratégies avec peu de perturbations comme $\text{Escape}_{\text{Mutation}}10\%$, que le seuil de détection doit être augmenté pour perturber la population plus tôt, car le mécanisme d'échappement ne semble pas assez puissant pour s'échapper suffisamment loin de l'optimum local.

Étude de la taille H de la fenêtre glissante. Nous allons maintenant nous intéresser aux profils de convergences pour étudier le paramètre H, en utilisant un seuil de détection $\tau \in \{0, 0.01\}$. On décide de ne pas étudier le seuil de détection $\tau = 0.1$ car les configurations avec ce seuil ne permettent pas d'obtenir des résultats de bonne qualité comme nous avons pu le constater précédemment.

Pour un seuil de détection $\tau = 0$, nous remarquons à travers la Figure 3.3 une différence significative entre les résultats de $H = 1$ et $H \in \{200, 500, 1000, 2000, 5000\}$. En général, utiliser uniquement l'historique de la dernière génération ($H = 1$) pour détecter un blocage avec un seuil $\tau = 0$ ne semble pas pertinent, sauf peut-être pour la stratégie $\text{Escape}_{\text{Mutation}}$ 10% avec l'instance $n = 50$ et $k = 4$. En effet, dans ce cas précis où la stratégie perturbe peu la population, nous avons vu précédemment qu'il semble important de perturber plus rapidement la population qu'avec les autres stratégies, ce qui peut être réalisé ici quand l'historique des améliorations est très petit. À l'exception de la valeur $H = 1$, les différences de performances obtenues ne sont pas significatives pour les instances de petite taille comme $n = 50$.

Considérons maintenant des instances de plus grande taille comme $n = 200$. Nous avions remarqué dans l'analyse générale de la section précédente qu'il semblait préférable de choisir de grande taille de fenêtre sous un budget de 10^7 évaluations. En s'intéressant à la Figure 3.3, on peut remarquer que ce n'est pas vrai pour des budgets plus faibles. En effet, des valeurs plus faibles du paramètre H donne de meilleurs résultats quand le nombre d'évaluations est plus faible. Par exemple, pour la stratégie d'échappement $\text{Escape}_{\text{Mutation}}$ 30%, l'algorithme est plus performant avec une valeur de $H \leq 500$ quand le budget est inférieur à environ 10^6 évaluations. Néanmoins, si on augmente le budget, l'algorithme devient plus performant avec des valeurs de $H \geq 5\,000$. Ce comportement semble se généraliser à chacune des instances et des stratégies d'échappement étudiées dans cette figure.

Si on augmente maintenant le seuil de détection à $\tau = 0.01$, on peut voir sur la

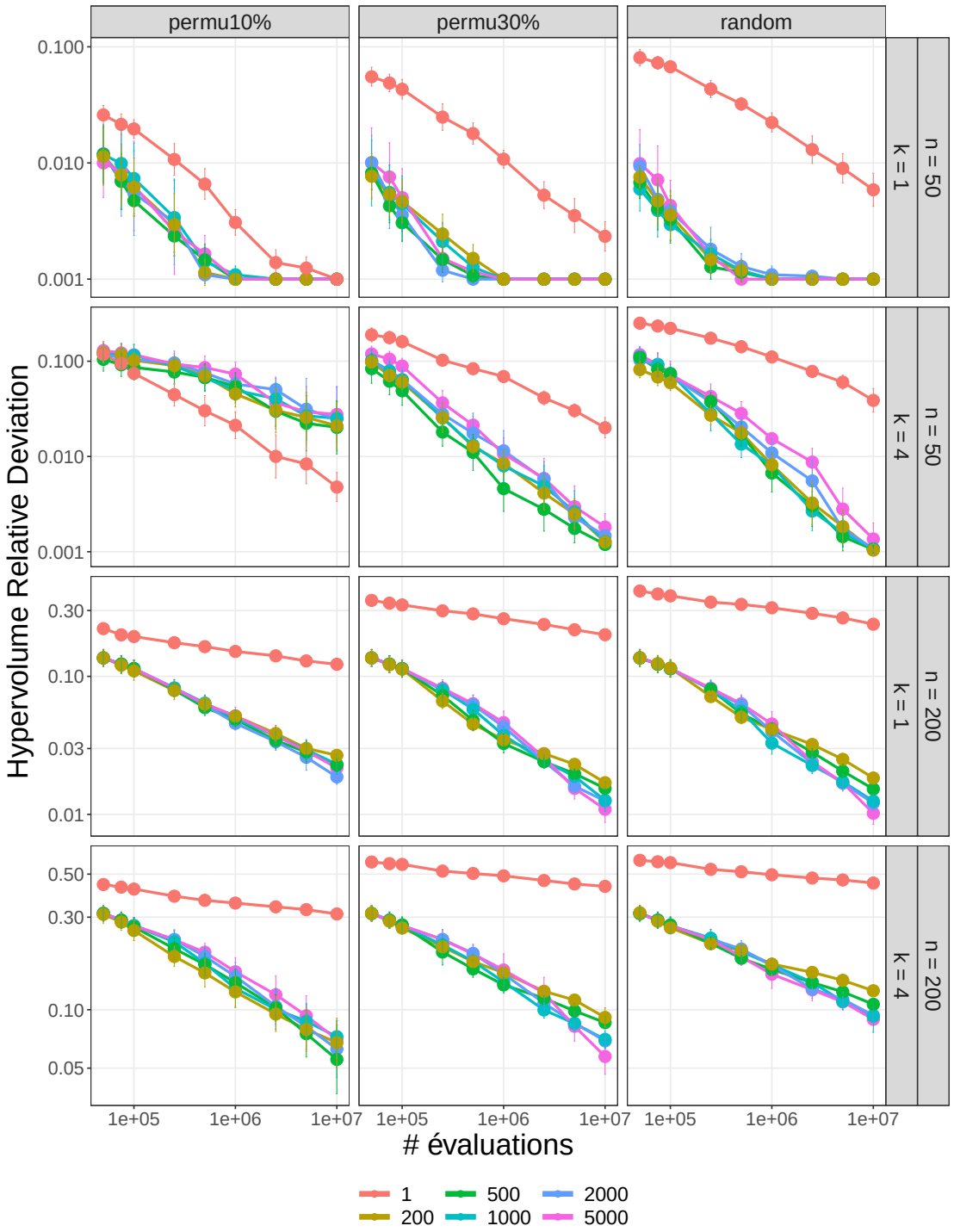

FIGURE 3.3 – Profils de convergence des stratégies d'échappement $\text{Escape}_{\text{Mutation}}$ (10% et 30%) et $\text{Escape}_{\text{Random}}$, avec un seuil $\tau = 0$ pour des tailles de fenêtres $H \in \{1, 200, 500, 1000, 2000, 5000\}$.

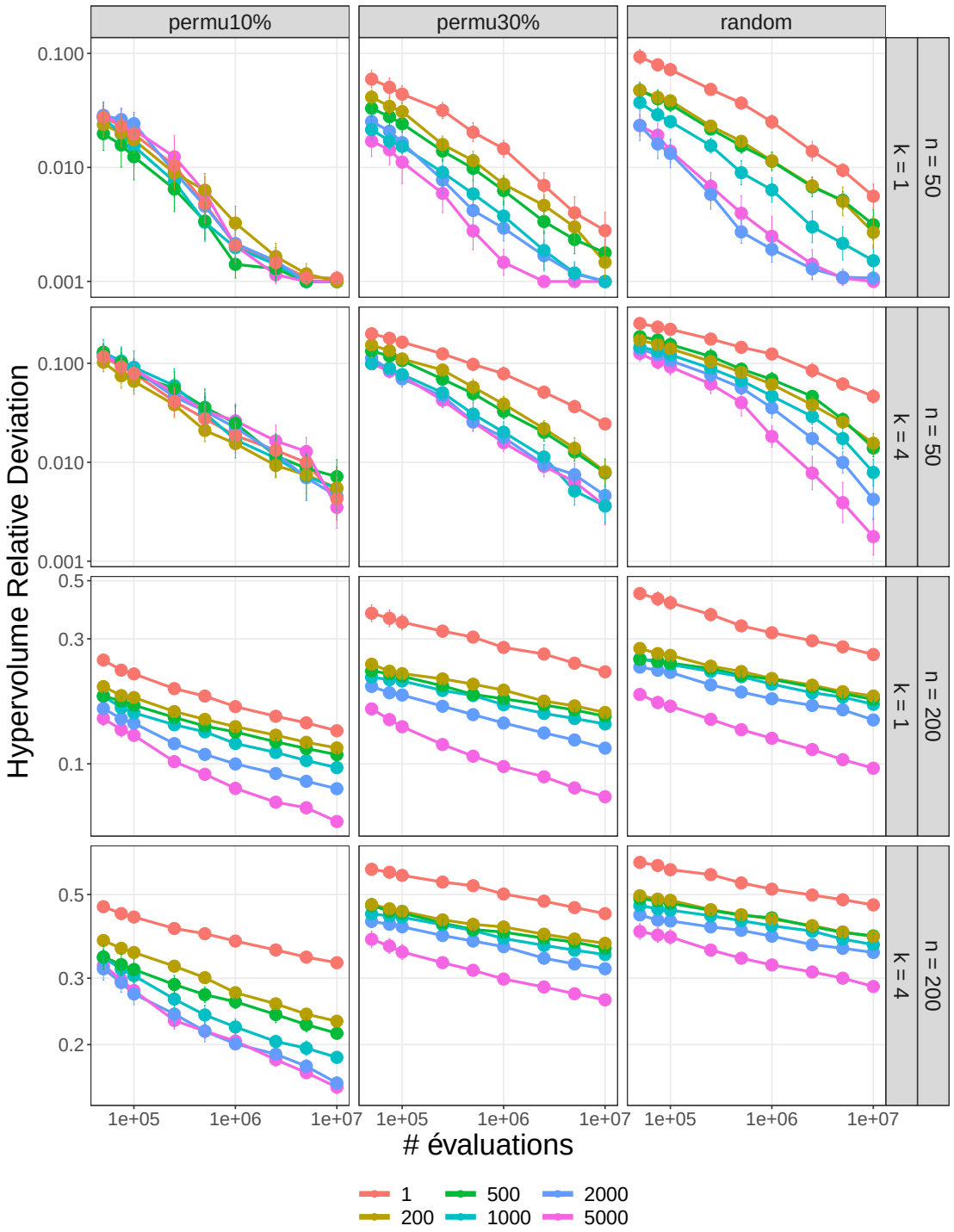

FIGURE 3.4 – Profils de convergence des stratégies d'échappement $\text{Escape}_{\text{Mutation}}$ (10% et 30%) et $\text{Escape}_{\text{Random}}$, avec un seuil $\tau = 0.01$ pour des tailles de fenêtres $H \in \{1, 200, 500, 1000, 2000, 5000\}$.

Figure 3.4 que les comportements des différentes configurations sont beaucoup plus stables qu'avec un seuil $\tau = 0$ (Figure 3.3). Pour des problèmes relativement petits ($n = 50$), la stratégie d'échappement $\mathsf{Escape}_{\mathsf{Mutation}}$ 10% semble privilégier des valeurs de H plus faibles, comme $H = 500$ pour l'instance $n = 50$ et $k = 1$, ou encore $H = 200$ avec l'instance $n = 50$ et $k = 4$. À part cette exception, toutes les autres configurations étudiées dans la Figure 3.4 nous montrent qu'il faut choisir la plus haute valeur de H pendant tout le processus de recherche. Cette observation complète et confirme les observations que l'on a pu faire dans la section précédente.

3.3.2.3 Résumé des observations

Pour détecter les blocages liés aux optima locaux, les paramètres τ et H, qui représentent respectivement le seuil de détection et la taille de l'historique qui stocke le nombre d'améliorations, doivent être configurés de manière rigoureuse pour ne pas dégrader les performances de MOEA/D. En effet, nous savons que des blocages occasionnels des sous-problèmes peuvent avoir lieu dans MOEA/D, ce qui peut avoir pour conséquence une erreur de détection, et donc une activation prématurée du processus d'échappement par perturbation. Le fait d'exécuter la stratégie d'échappement trop tôt va alors perturber la population avant qu'elle n'ait atteint un optimum local. On peut exécuter la stratégie d'échappement trop tôt si le seuil de détection est trop élevé ou si la taille de l'historique H est trop faible. Le seuil de détection le plus bas nous a donné les meilleurs résultats pour les instances étudiées. Dans la suite, nous allons utiliser un seuil de détection à $\tau = 0$.

D'un autre côté, le fait de ne pas détecter le blocage assez tôt peut entraîner la perte de ressources de calcul en ne s'échappant pas assez rapidement. En effet, on a pu remarquer sur l'instance de mnk-landscapes avec $n = 50$ et $k = 4$, que si l'on a un historique trop important avec un seuil faible, alors on ne peut pas obtenir des résultats optimaux. On a pu remarquer également que la meilleure valeur à adopter pour configurer la taille de l'historique était dépendante du budget disponible. En effet, de manière générale, les meilleurs résultats sont obtenus avec des valeurs élevées de H, sauf durant les premières évaluations où des valeurs plus faibles donnent de meilleurs résultats.

3.3.3 Étude des stratégies de perturbations

Pour étudier plus facilement les différentes stratégies d'échappement aux optima locaux, nous avons choisi de fixer le seuil de détection τ et la taille de l'historique H

avec les valeurs qui nous ont permis d'obtenir les meilleurs résultats dans la section précédente. Nous avons donc choisi pour l'ensemble des configurations un seuil de détection $\tau = 0$ et une taille d'historique $H = 2\,000$, sauf pour la stratégie d'échappement $\text{Escape}_{\text{Mutation}}$ 10% pour les instances avec $k = 4$ et $n \in \{50, 100\}$ où nous avons choisi $\tau = 0.01$ et $H = 5\,000$. On notera ici que l'utilisation de ces paramètres ne nous garantit pas de trouver les résultats optimaux, mais nous permettent de trouver de bons résultats pour chacune des instances étudiées. Nous analysons alors les stratégies d'échappement pour les deux stratégies de sélection des sous problèmes, sps_{All} et sps_{Rnd}, que nous avons considérés dans le chapitre précédent. En effet, comme on a pu le voir dans le chapitre précédent, la stratégie sps_{Rnd} améliore le profil de convergence de l'algorithme en ayant toutefois quelques limitations une fois proche du front Pareto, ou lorsque le degré d'épistasie est particulièrement élevé.

Pour analyser l'impact des différentes stratégies d'échappement implémentées dans cette étude, nous nous appuierons sur le Tableau 3.5 ainsi que la Figure 3.5. Le tableau montre les rangs de chacune des configurations possibles sous un budget de 10^7 évaluations. Les configurations sont confrontées les unes avec les autres pour une stratégie de sélection des sous-problèmes (*sps*) donnée pour toutes les instances de mnk-landscapes considérées. Dans ce tableau, les stratégies d'échappement sont classées par degré de perturbation croissante. Autrement dit, la première stratégie $\text{Escape}_{\text{Mutation}}$ 10% est la stratégie qui perturbe le moins la population alors que la stratégie d'échappement $\text{Escape}_{\text{Random}}$ est celle qui la perturbe le plus. La Figure 3.5 étudient les profils de convergences des stratégies d'échappement dans le même contexte que le Tableau 3.5.

Étude globale des différentes stratégies après 10^7 évaluations. D'un point de vue globale, deux observations peuvent être faites sur le Tableau 3.5. La première chose que l'on remarque est que chaque configuration étudiée sur l'instance $n = 50$ et $k = 1$ arrive à converger avant 10^7 évaluations ce qui ne nous permet pas de tirer des conclusions pour le moment. La seconde observation que l'on peut faire concerne le comportement des deux stratégies de sélection des sous-problèmes sps_{All} et sps_{Rnd}. En effet, il est très clair que leur comportement est pratiquement identique pour chacune des instances étudiées et des mécanismes d'échappement utilisés, même si les valeurs de la déviation relative de l'hypervolume sont meilleures pour sps_{Rnd}.

Intéressons-nous maintenant aux instances ayant un degré d'épistasie assez faible ($k = 1$). Pour rappel, augmenter le degré d'épistasie a pour effet d'augmenter le nombre d'optima locaux et inversement si on diminue le degré d'épistasie. Pour ce type de problèmes, la stratégie d'échappement qui s'en sort le mieux est la stratégie $\text{Escape}_{\text{Random}}$,

Tableau 3.5 – Rangs et moyenne (entre parenthèse) de la déviation relative d'hypervolume, obtenue après 10^7 évaluations pour les stratégies d'échappement $\mathsf{Escape}_{\mathsf{Mutation}}10\%$, $\mathsf{Escape}_{\mathsf{Mutation}}20\%$, $\mathsf{Escape}_{\mathsf{Mutation}}30\%$ et $\mathsf{Escape}_{\mathsf{Random}}$. Chacune des stratégies d'échappement sont comparés une à une, indépendamment des stratégies de sélection $\mathsf{sps}_{\mathrm{All}}$ et $\mathsf{sps}_{\mathrm{Rnd}}$. Les valeurs d'hypervolume soulignées correspondent à la meilleure valeur moyenne.

		$k=1$			$k=4$		
		$n=50$	$n=100$	$n=200$	$n=50$	$n=100$	$n=200$
$\mathsf{sps}_{\mathrm{All}}$	$\mathsf{Escape}_{\mathsf{Mutation}}10\%$	$0_{(\underline{0.1})}$	$3_{(0.783)}$	$3_{(1.902)}$	$3_{(0.46)}$	$2_{(3.298)}$	$0_{(6.953)}$
	$\mathsf{Escape}_{\mathsf{Mutation}}20\%$	$0_{(\underline{0.1})}$	$2_{(0.555)}$	$0_{(1.483)}$	$2_{(0.247)}$	$0_{(\underline{1.586})}$	$0_{(\underline{4.797})}$
	$\mathsf{Escape}_{\mathsf{Mutation}}30\%$	$0_{(\underline{0.1})}$	$1_{(0.36)}$	$0_{(1.274)}$	$1_{(0.15)}$	$1_{(2.172)}$	$1_{(6.98)}$
	$\mathsf{Escape}_{\mathsf{Random}}$	$0_{(\underline{0.1})}$	$0_{(\underline{0.238})}$	$0_{(\underline{1.229})}$	$0_{(\underline{0.109})}$	$1_{(2.826)}$	$3_{(9.436)}$
$\mathsf{sps}_{\mathrm{Rnd}}$	$\mathsf{Escape}_{\mathsf{Mutation}}10\%$	$0_{(\underline{0.1})}$	$3_{(0.425)}$	$3_{(0.65)}$	$3_{(0.998)}$	$2_{(2.384)}$	$0_{(\underline{2.281})}$
	$\mathsf{Escape}_{\mathsf{Mutation}}20\%$	$0_{(\underline{0.1})}$	$1_{(0.19)}$	$0_{(0.392)}$	$2_{(0.159)}$	$0_{(\underline{0.731})}$	$0_{(2.908)}$
	$\mathsf{Escape}_{\mathsf{Mutation}}30\%$	$0_{(\underline{0.1})}$	$1_{(0.152)}$	$0_{(\underline{0.382})}$	$0_{(\underline{0.115})}$	$1_{(1.31)}$	$2_{(6.415)}$
	$\mathsf{Escape}_{\mathsf{Random}}$	$0_{(\underline{0.1})}$	$0_{(\underline{0.121})}$	$0_{(0.448)}$	$0_{(0.121)}$	$2_{(2.369)}$	$3_{(8.075)}$

qui effectue la perturbation la plus radicale dans la population. Le fait d'avoir un paysage assez lisse rend l'échappement aux optima plus compliqué pour les perturbations les plus faibles qui n'arrivent pas à s'échapper de l'optimum local. Cette observation semble s'atténuer quand la taille de l'espace de décision devient plus importante, en passant de $n = 100$ à $n = 200$ par exemple.

Intéressons-nous maintenant aux instances de mnk-landscapes avec un paysage plus rugueux ($k = 4$). Si on reprend notre raisonnement pour l'étude des instances avec $k = 1$, on peut alors s'attendre à ce que les meilleurs résultats soient obtenus par les stratégies qui perturbent le moins la population. On remarque en effet ce comportement pour des instances avec un espace de décision assez grand comme $n = 200$, où les meilleurs résultats sont obtenus par les stratégies $\mathsf{Escape}_{\mathsf{Mutation}}$ 10% et $\mathsf{Escape}_{\mathsf{Mutation}}$ 20%. Cependant, ce comportement semble s'inverser au fur et à mesure que la taille de l'espace de décision diminue. En effet, la meilleure stratégie est $\mathsf{Escape}_{\mathsf{Mutation}}$ 20% pour les instances avec $n = 100$ et $\mathsf{Escape}_{\mathsf{Random}}$ pour les instances les plus petites avec $n = 50$.

Étude des profils de convergence des stratégies d'échappement. Étudions maintenant les profils de convergences des différentes stratégies d'échappement aux optima locaux sur la Figure 3.5. On peut voir très clairement que l'utilisation d'une stratégie d'échappement est très bénéfique pour améliorer les performances de MOEA/D, que ça soit avec $\mathsf{sps}_{\mathrm{All}}$ ou $\mathsf{sps}_{\mathrm{Rnd}}$. Dans le pire des cas, quand la stratégie utilisée n'est pas adaptée, comme la stratégie $\mathsf{Escape}_{\mathsf{Mutation}}$ 10% pour l'instance avec $n = 100$ et $k = 1$, les

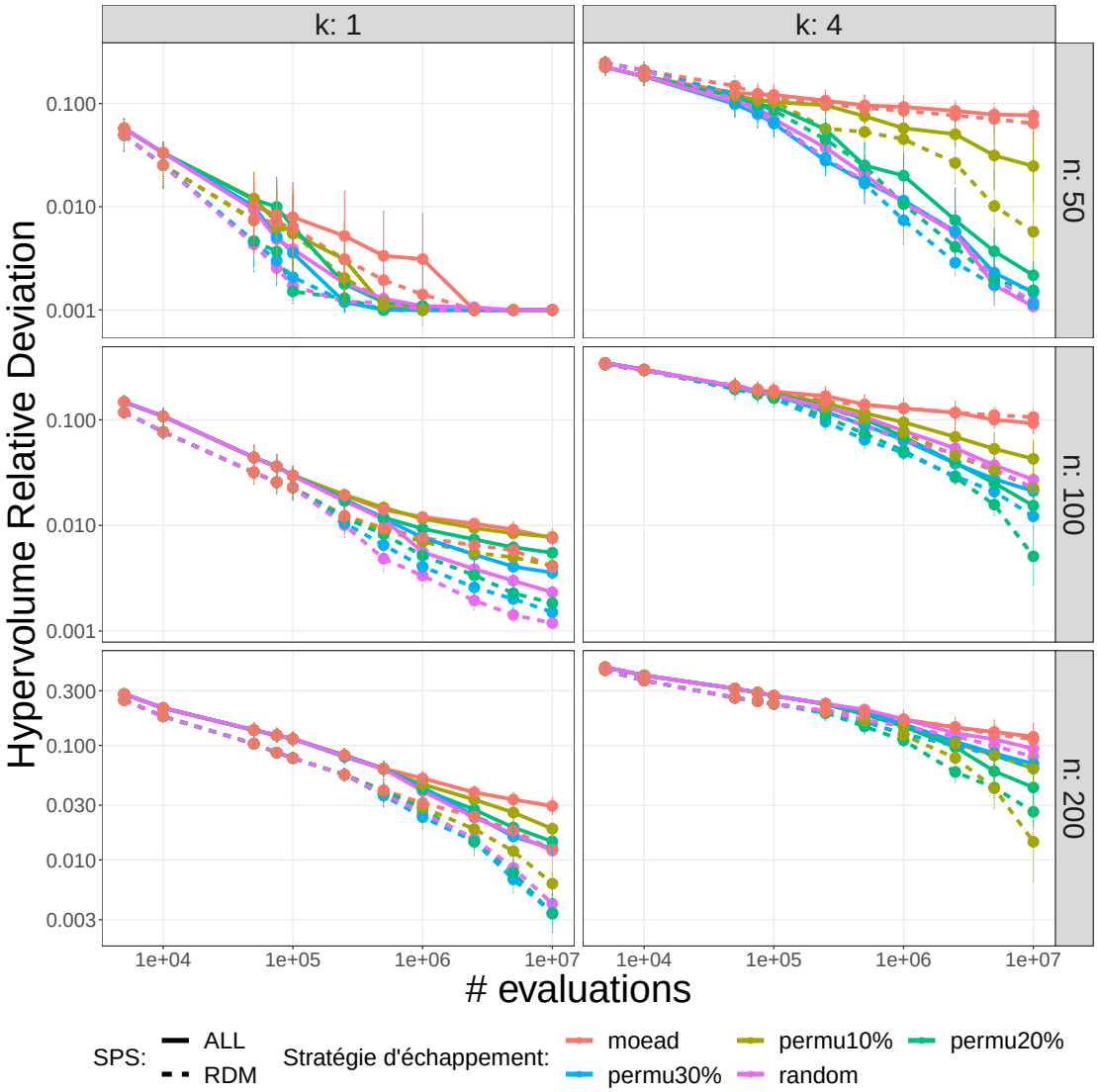

FIGURE 3.5 – Profils de convergence des différentes stratégies d'échappement avec une taille de fenêtre $H = 2\,000$ et un seuil $\tau = 0$.

performances ne seront pas dégradées si les paramètres de détection des optima locaux sont bien configurés. En effet, si on détecte correctement le blocage du processus de recherche à cause d'un optimum local, le fait que la stratégie d'échappement utilisée ne parvienne pas à s'échapper de l'optimum local donnera, dans le pire des cas, les mêmes performances que MOEA/D sans échappement.

Continuons maintenant à nous intéresser à MOEA/D sans stratégie d'échappement, mais avec la stratégie de sélection des sous-problèmes sps_{RND}. On peut alors noter

que l'algorithme avec sps_{RND} améliore significativement MOEA/D avec la stratégie sps_{ALL} pour les problèmes lisses ($k = 1$), mais il a plus de difficultés quand le problème est rugueux ($k = 4$) comme cela a déjà pu être mis en avant de façon détaillée dans le chapitre précédent. Pour rappel, on avait remarqué que parcourir un seul sous-problème aléatoirement à chaque génération avait un intérêt de plus en plus limité quand le problème devenait de plus en plus rugueux. On peut toutefois remarquer sur la Figure 3.5 que l'utilisation des stratégies d'échappement en complément de la stratégie sps_{RND} améliore considérablement les performances de MOEA/D avec sps_{RND}. Le fait de permettre à MOEA/D de s'échapper des optima locaux favorise la stratégie sps_{RND} à exploiter au maximum ses capacités en obtenant les meilleures performances pour la quasi-totalité des instances étudiées.

3.4 Conclusion

Dans ce chapitre, nous avons étudié la pertinence de l'utilisation d'un système d'échappement dans MOEA/D. Nous avons notamment proposé un mécanisme de détection, ainsi que différentes stratégies de perturbation de la population. On a pu voir que le choix de la stratégie d'échappement optimale se fait en fonction du problème à optimiser, et plus précisément du nombre d'optima locaux présents dans l'instance du problème. En outre, un bon mécanisme de détection est primordial pour activer la stratégie d'échappement au moment opportun. Notre étude a révélé que le paramétrage du seuil de détection et de la taille de l'historique influent énormément sur les performances de l'algorithme. On a pu voir par exemple que l'on peut obtenir de bons résultats pour une instance en perturbant fortement la population, mais que l'on peut tout aussi bien obtenir de mauvais résultats pour la même instance et la même stratégie d'échappement lorsque les paramètres de détection ne sont pas correctement configurés. Avec le système de détection que nous utilisons, si le seuil de détection est trop haut, ou si la taille de l'historique est trop basse, il se peut que l'on détecte un blocage temporaire qui soit trompeur, et qui peut entraîner la perturbation d'une population qui est en cours de convergence, ce qui peut par conséquent ralentir la convergence de MOEA/D. Le paramétrage de la taille de l'historique semble quant à lui être dépendant du budget disponible. Il serait alors intéressant d'approfondir cette observation en proposant un paramétrage adaptatif de la taille de l'historique pour avoir un historique assez petit au début du processus de recherche, et plus important vers la fin.

Finalement, on a pu voir que l'utilisation d'un système d'échappement dans MOEA/D permet d'améliorer les performances de l'algorithme, peu importe la stratégie de sé-

lection des sous-problèmes utilisée. Le système d'échappement semble même enlever les limites de la stratégie $\mathrm{sps}_{\mathrm{RND}}$ que l'on avait pu apercevoir dans le Chapitre 2. Pour rappel, l'utilisation de MOEA/D-($\mu = 500$, $\lambda = 1$, $\mathrm{sps}_{\mathrm{RND}}$) améliore significativement le profil de convergence de l'algorithme, mais semblait toutefois limitée pour des degrés d'épistasie élevés des instances mnk-landscapes. L'utilisation combinée de la stratégie $\mathrm{sps}_{\mathrm{RND}}$ et d'un système d'échappement permet de montrer que cette limitation venait de la rugosité du paysage, et que l'on peut désormais combler cette faiblesse en combinant ces deux composants.

Chapitre 4

Optimisation coûteuse par décomposition assistée par méta-modèle

Résumé

Dans ce chapitre, nous présentons la dernière contribution de cette thèse, qui consiste à prendre en compte la nature coûteuse de certains problèmes d'optimisation. Dans un contexte d'optimisation coûteuse, le budget disponible en termes du nombre d'appels à la fonction d'évaluation est extrêmement restreint. En conséquence, il est indispensable de pouvoir assister le processus d'optimisation en utilisant des méta-modèles permettant de guider la recherche vers des solutions de bonne qualité, tout en réduisant le nombre d'appels à la fonction d'évaluation coûteuse. Il s'agit là d'une problématique le plus souvent étudiée pour des problèmes continus et/ou mono-objectif. Dans nos travaux, nous proposons d'étudier cette problématique dans le contexte de problèmes d'optimisation à la fois combinatoires et multi-objectifs. Pour cela, nous proposons un framework générique basé sur le concept de décomposition introduit dans MOEA/D afin de guider la recherche, ainsi que sur l'intégration de modèles de substitution combinatoires basés sur la décomposition de Walsh. Nous présentons également une étude empirique extensive des différents composants qui constituent notre framework en démontrant de façon approfondie leur impact sur le processus d'optimisation multi-objectif. Une partie de ces travaux ont fait l'objet d'une publication à la conférence internationale GECCO 2020 [83], ainsi que d'une extension en cours de soumission [84].

Dans ce chapitre, nous commençons par introduire dans la Section 4.1 les concepts

clés liés à l'optimisation coûteuse. Nous donnons ensuite dans la Section 4.2 une présentation détaillée de notre framework à base de décomposition pour l'optimisation multi-objectif combinatoire coûteuse. Ce framework et ses différents composants sont étudiés dans l'analyse expérimentale de la Section 4.3, avant de conclure le chapitre dans la Section 4.4.

4.1 Optimisation multi-objectif coûteuse

4.1.1 Contexte et motivations

L'optimisation multi-objectif trouve des applications dans des domaines complexes introduisant des contraintes fortes sur le nombre total de solutions que l'on peut examiner tout au long du processus d'optimisation. En ingénierie par exemple, on a souvent recours à des simulateurs pour évaluer la qualité d'une solution. Ainsi, évaluer une solution peut être sujet à des opérations complexes, très gourmandes en temps de calcul. L'optimisation coûteuse fait référence à la résolution de problèmes dont le ou les objectifs impliquent un coût de calcul très important, allant de plusieurs secondes à plusieurs heures. Dans ce contexte, le but est donc de pouvoir résoudre au mieux le problème considéré en utilisant le moins possible d'appels à la fonction d'évaluation, car cela domine par définition le coût global du processus d'optimisation.

Pour des problèmes d'optimisation coûteuse, les contraintes en termes de temps de calcul sont si fortes qu'il devient impossible en pratique d'utiliser des méta-heuristiques et des algorithmes évolutionnaires dans leurs versions classiques. En effet, il est bien connu que ces algorithmes, bien qu'extrêmement efficaces pour trouver des solutions de bonne qualité pour de nombreux problèmes et applications, peuvent nécessiter un nombre relativement élevé d'appels à la fonction d'évaluation. À titre d'exemple, en examinant les profils de convergence présentés dans les chapitres précédents, on peut remarquer que MOEA/D nécessite au moins 10^5 évaluations pour s'approcher du front Pareto pour les instances les plus simples du problème mnk-landscapes ($m = 2$, $n = 100$ et $k = 0$) et plus de 10^7 évaluations sur les instances les plus compliqués ($m = 5$, $n = 100$ et $k = 4$). En supposant que le temps d'exécution d'une seule évaluation prendrait 10 secondes, ce qui reste raisonnable dans le contexte de l'optimisation coûteuse, il faudrait au moins 11.5 jours pour exécuter une seule fois MOEA/D.

Il devient donc évident que le défi majeur en optimisation coûteuse est de concevoir un algorithme pouvant produire des solutions de bonne qualité en utilisant très peu d'appels à la fonction d'évaluations, typiquement, de l'ordre de quelques centaines, ou

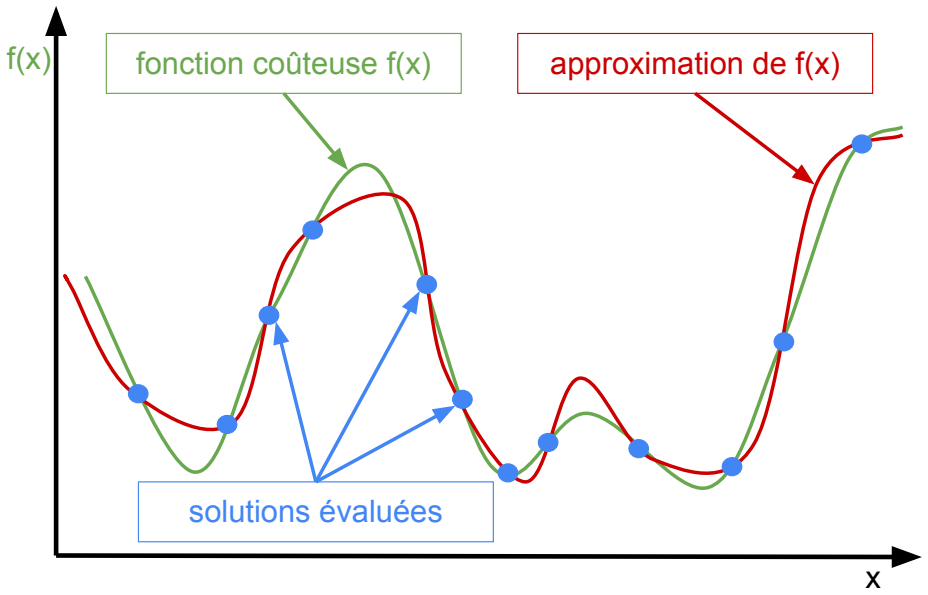

FIGURE 4.1 – Représentation graphique d'un méta-modèle

au plus quelques milliers, d'évaluations. Pour cela nous nous intéressons à l'utilisation de ce qu'on appelle des modèles de substitution, aussi appelé méta-modèles, pour assister le processus de recherche évolutionnaire. En effet, dans le cadre de nos travaux, l'utilisation d'un méta-modèle est principalement vue comme étant un moyen alternatif pour évaluer (de façon approximative) la qualité d'une solution candidate sans faire appel à la fonction d'évaluation coûteuse. Cela est discuté un peu plus en détail dans les sections suivantes.

4.1.1.1 Méta-modèles ou modèles de substitution

Dans nos travaux, un méta-modèle peut être vu comme un substitut à une fonction objectif coûteuse. Il s'agit en effet d'*apprendre* un modèle particulier en utilisant des solutions préalablement évaluées pour ensuite donner une approximation de la fonction coûteuse. La Figure 4.1 représente un méta-modèle (la fonction en rouge) qui permet de donner une approximation de la fonction coûteuse $f(x)$ (en vert). L'intérêt d'utiliser un méta-modèle est que l'approximation donnée par le méta-modèle est beaucoup moins coûteuse que la fonction initiale.

Afin de mieux illustrer notre discussion, considérons à titre d'exemple le cas typique, et largement étudié dans la littérature, d'un modèle linéaire simple. Étant donné

un espace de recherche à variables continues $\{x_1, x_2, ..., x_n\}$, et une fonction coûteuse $f : \mathbb{R}^n \to R$, on cherche alors à apprendre une fonction linéaire $\tilde{f}(x)$ qui approxime la fonction f, et qui s'écrit sous la forme :

$$\tilde{f}(x) = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + ... + \omega_n x_n \tag{4.1}$$

Dans une première étape, il s'agit donc de calculer les coefficients $\{\omega_0, \omega_1, \omega_2, ..., \omega_n\}$ afin d'avoir une approximation $\tilde{f}$ la plus proche possible de la fonction f. Une fois les coefficients du méta-modèle $\tilde{f}$ estimés, on peut alors utiliser $\tilde{f}$ comme un substitut de f, pour optimiser le problème dont la fonction objectif est spécifiquement la fonction coûteuse f. Il est donc clair que dans le cadre d'un algorithme assisté par méta-modèle, il est nécessaire de définir un méta-modèle capable d'approximer la fonction coûteuse, et d'être en mesure d'apprendre (les coefficients de) ce méta-modèle tout au long du processus d'optimisation, ceci à partir d'un ensemble d'apprentissage constitué de solutions déjà évaluées. En particulier, à part le choix du méta-modèle (qui sera discuté plus loin dans ce chapitre), deux aspects importants sont à prendre en compte :

— La constitution de l'ensemble d'apprentissage permettant d'inférer les coefficients du modèle. L'approche la plus simple consiste à ajouter les solutions évaluées au fur et à mesure de la progression de l'algorithme d'optimisation à l'ensemble d'apprentissage. Cette approche n'est pas toujours envisageable en pratique, notamment à cause d'un temps d'apprentissage du méta-modèle qui peut parfois s'avérer extrêmement long. Néanmoins, dans nos travaux, nous considérons que le temps d'apprentissage est toujours dominé par le coût d'évaluation de la fonction à optimiser.

— L'algorithme permettant d'entraîner le méta-modèle, c'est-à-dire la détermination des (hyper-)paramètres inhérents au modèle considéré. Par exemple, dans le cas d'un modèle linéaire simple, différents algorithmes existent pour calculer les coefficients de $\tilde{f}$, tels que Lasso [34] ou Lars [26]. Le choix d'un algorithme particulier peut jouer un rôle déterminant dans la qualité du méta-modèle obtenu. Dans nos travaux, nous nous appuierons sur les algorithmes existants de la littérature, bien qu'il s'agisse ici d'un problème d'optimisation en soi.

4.1.1.2 Optimisation assistée par méta-modèle

Comme mentionnée ci-dessus, nous allons faire l'hypothèse que le méta-modèle ainsi calculé permet de donner une approximation de l'évaluation d'une solution, de façon bien moins coûteuse que la fonction originale. Différentes approches permettent

Algorithme 7 : Méthodologie d'optimisation coûteuse

```
1 D ← base d'apprentissage initiale du modèle;
2 tant que le critère d'arrêt n'est pas atteint faire
      // Calcul des hyper-paramètres du méta-modèle
3     M ← apprentissage du modèle avec la base D;
      // Calcul d'une solution sur la base du méta-modèle
4     x ← meta-heuristique(M);
      // Ajout de la nouvelle solution évaluée avec la fonction
         coûteuse dans la base d'apprentissage
5     D ← D ∪ (x, f(x))
```

alors d'intégrer un méta-modèle au sein d'un algorithme d'optimisation évolutionnaire afin d'assister le processus de recherche. En optimisation mono-objectif, une approche basique et souvent adoptée est résumée dans le template de l'Algorithme 7. L'algorithme se base dans un premier temps sur le méta-modèle, en tant que substitut, pour trouver une solution potentielle x. Cette solution est ensuite évaluée de façon coûteuse, avant d'être ajoutée à l'ensemble d'apprentissage. Le modèle d'apprentissage est par conséquent de nouveau construit, et le processus est itéré jusqu'à épuisement du budget disponible. Cette approche peut être généralisée à l'optimisation multi-objectif. Cependant, plusieurs questions se posent. En particulier, en optimisation multi-objectif, le but est de trouver un ensemble de solutions et non pas une solution unique. Il devient alors critique de déterminer quelles solutions calculer, et comment, en utilisant les méta-modèles relatifs aux objectifs. La contribution décrite dans ce chapitre reviendra plus précisément sur les différents composants permettant d'adapter le template de l'Algorithme 7 pour l'optimisation multi-objectif. Pour l'instant, nous proposons de commencer par une revue de l'état-de-l'art des méta-modèles pour les problèmes à variables discrètes.

4.1.2 Méta-modèles pour l'optimisation combinatoire coûteuse

La plupart des travaux de la littérature traitent de méta-modèles pour l'optimisation continue. En effet, la plupart des techniques inhérentes aux modèles de substitutions trouvent leurs fondations premières dans le cadre de fonctions à variables continues. Cependant, on peut trouver des études récentes consacrées au développement de méta-modèles pour des problèmes combinatoire [5, 6, 57, 50, 113], et en particulier pour des fonctions à variables binaires [89]. Nous les décrivons brièvement dans les sections suivantes.

4.1.2.1 Méta-modèle basé sur les fonctions à base radiale (RBF)

Une fonction à base radiale (RBF) [37, 36] est une fonction réelle dont la valeur dépend de la distance entre un point c, appelé *centre*, et qui peut être notée : $\phi(x) = \phi(dist(x,c))$, où $dist(x,y)$ est une distance entre deux solutions x et y. Les fonctions RBF sont utilisées pour donner des fonctions d'approximation de la forme suivante :

$$\tilde{f}(x) = w_0 + \sum_{i=1}^{N} w_i \exp\left(-\beta \text{ dist } (x, c_i)^2\right) \tag{4.2}$$

L'approximation de la fonction f est ainsi donnée par la fonction $\tilde{f}$ représentée par la somme de N fonctions à base radiale, chacune associée à un centre c_i différent et pondéré par un coefficient w_i différent. Un biais w_0 est également ajouté. La constante β est définie par $\beta = 1/2\ D^2$, avec D la distance maximale. Il a été montré que grâce à la méthodologie de généralisation géométrique [71, 110], toutes les fonctions peuvent donner lieu à une bonne approximation grâce à cette équation sous certaines conditions.

4.1.2.2 Méta-modèle avec techniques de Krigeage

Dans les travaux de Moraglio et Kattan, traitant de l'utilisation de modèles RBF pour les espaces combinatoires [72], les auteurs suggèrent d'utiliser des modèles plus complexes comme les processus gaussiens. En effet, les modèles basés sur les processus gaussiens [113, 112], aussi appelés krigeage, permettent de calculer une fonction d'acquisition appelée *amélioration espérée* (ou *Expected Improvement*, EI) en utilisant l'erreur estimée de chaque prédiction qui est disponible avec le krigeage. Cette fonction d'acquisition est notamment utilisée dans le framework d'optimisation EGO (*Efficient Global Optimization*) [88] pour choisir les solutions qui doivent être évaluées avec la vraie fonction d'évaluation. Pour adapter les modèles basés sur les techniques de krigeage, la distance euclidienne est remplacée par la distance de Hamming comme pour les modèles RBF. L'approximation de la fonction f est alors donnée par le processus gaussien suivant :

$$\tilde{f}(x) = \mathcal{N}(\mu(x), \sigma(x)) \tag{4.3}$$

avec la moyenne μ représentée par l'espérance des solutions dans la base d'apprentissage $\mu(x) = E[f(x)]$ et la matrice de covariance représentée par $\sigma(x) = \exp(-\theta \text{ dist } (x,x')^p)$ où la distance est représentée par la distance de Hamming dans un contexte combinatoire binaire. Les paramètres θ et p sont des réels à estimer. Les travaux de Zaefferer [111, 113] ont montré que ce modèle surpasse le modèle basé sur les fonctions radiales en

optimisation mono-objectif combinatoire.

4.1.2.3 Méta-modèle bayésien

BOCS (*Bayesian Optimization for Combinatorial Structures*) [5] est un algorithme pour l'optimisation assistée par méta-modèle qui s'appuie sur un modèle bayésien ayant pour but de capturer et d'utiliser les interactions entre les variables. Plus précisément, il est spécifique aux problèmes à variables booléennes et utilise un modèle polynomial multi-linéaire de la forme suivante :

$$\tilde{f}(x) = \sum_{T \subset N} a_T \prod_{i \in T} x_i \tag{4.4}$$

avec $N \in \{1, \ldots, n\}$, x_i la i^{eme} variable de la solution binaire x. On remarque que l'équation précédente se présente sous la forme d'un polynôme où chaque monôme est pondéré par un coefficient a_T. Ce modèle est inutilisable dans sa forme la plus générale à cause du nombre exponentiel de monômes qui peuvent être présents. De ce fait, le nombre de monômes est en pratique limité par un ordre k. Un ordre plus grand rend le modèle plus expressif, mais diminue la précision si le nombre de solutions dans la base d'apprentissage est trop limité. Dans [5], les auteurs considèrent que l'ordre 2 est un bon compromis en pratique pour les problèmes quadratiques à faible dimension ($n < 25$).

4.1.2.4 Méta-modèle basé sur les fonctions de Walsh

Les fonctions de Walsh [98], appelées ainsi d'après Joseph L. Walsh, constituent un ensemble de fonctions ϕ_k pouvant être utilisées pour concevoir des méta-modèles pour les fonctions pseudo-booléennes. Les fonctions de Walsh composent une base normale et orthogonale qui peut être utilisée pour décomposer les fonctions de l'espace de Hilbert sous certaines conditions [98, 25], comme on le ferait dans une transformée de Fourrier.

Ces fonctions prennent en paramètre des chaînes binaires et ont pour valeur de retour deux valeurs possibles : -1 ou 1. Plus précisément, pour tout entier k avec la représentation binaire $k = k_0 + 2k_1 + \ldots + 2^m k_m$ où m est un entier avec $k_i \in \{0, 1\}$, on peut définir la k^{ieme} fonction de Walsh $\phi_k : 0, 1^n \rightarrow \{-1, 1\}$ pour toute chaîne binaire $x = (x_1, \ldots, x_i, \ldots, x_n) \in \{0, 1\}^n$ de taille n de la façon suivante [98] :

$$\phi_k(x) = (-1)^{\sum_{i-0}^{n-1} k_i x_i} \tag{4.5}$$

L'ordre de la fonction de Walsh ϕ_k, noté $o(\phi_k)$, est défini par le nombre de bits k_i à 0

dans la représentation binaire de k. On peut ainsi dire qu'il existe une seule fonction d'ordre 1 qui est la fonction ϕ_0. Les fonctions de Walsh d'ordre 2 sont les fonctions ϕ_{2^p} pour tout entier $p \geqslant 0$ et les fonctions d'ordre 3 sont les fonctions $\phi_{2^p+2^{p'}}$ pour toutes les paires d'entiers $p \neq p' \geqslant 0$.

Avec ces définitions, il n'est pas difficile de montrer que l'ensemble des fonctions de Walsh forment une base normale orthogonale pour l'espace des fonctions booléennes, c'est-à dire que $\forall k, k' \in [\![0, 2^n - 1]\!]$, on a alors :

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} \phi_k(x) \cdot \phi_{k'}(x) = \delta_{kk'} \tag{4.6}$$

Ceci rend la base de Walsh extrêmement intéressante pour représenter des fonctions pseudo-booléennes de façon unique. En effet, toute fonction $f : \{0,1\}^n \to \mathbb{R}$ peut être décomposée sous la forme suivante :

$$\forall x \in \{0,1\}^n, f(x) = \sum_{k=0}^{2^n-1} w_k \cdot \phi_k(x) \tag{4.7}$$

avec chaque coefficient w_k défini de la façon suivante :

$$w_k = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x) \cdot \phi_k(x) \tag{4.8}$$

Dans l'équation (4.7), il est important de remarquer que la décomposition de Walsh d'une fonction est composée de deux éléments : les fonctions de Walsh et les coefficients. Les fonctions de Walsh sont définies indépendamment de la fonction f, comme on peut le voir dans l'équation (4.6). La définition des coefficients w_k dépend quant à elle de la fonction f comme on le voit avec l'équation (4.8). Notons que chaque coefficient correspond à une fonction de Walsh d'un ordre donné k, permettant de capturer l'interaction entre un ensemble de k variables au sein de la fonction f. On peut par conséquent espérer pour des fonctions avec peu d'interactions entre les variables, que le nombre de coefficients égaux à zéro est élevé, c'est-à-dire que seul un nombre relativement limité de coefficients non-nuls existe. C'est typiquement le cas pour des problèmes combinatoires comme $MUBQP$, réputé par ailleurs difficile, puisqu'il s'agit d'un problème quadratique par définition. C'est aussi le cas pour les instances du problème mnk-landscapes avec des valeurs fixées du paramètre k.

En se basant sur cette observation, l'idée développée dans [98, 57] est de permettre une approximation de la fonction f en utilisant les fonctions de Walsh limitées à

un certain ordre (de faible valeur), et en calculant une approximation $\hat{w_k}$ des vrais coefficients w_k dans la décomposition exacte de Walsh. Plus formellement, un méta-modèle basé sur les fonctions de Walsh peut être défini de la façon suivante :

$$\tilde{f}(x \mid d) = \sum_{k:o(\phi_k)\leqslant d} \tilde{w}_k \cdot \phi_k(x) \tag{4.9}$$

pour un ordre donné $d \ll n$. En particulier, l'approximation de la fonction f pour un ordre des fonctions de Walsh limité à 2 peut être écrite de la façon suivante :

$$\tilde{f}(x \mid 2) = \bar{w}_0 + \sum_{i=1}^{n} \bar{w}_i \cdot (-1)^{x_i} + \sum_{i<j<n} \bar{w}_{ij} \cdot (-1)^{x_i + x_j} \tag{4.10}$$

où $\bar{w}_0$ est le coefficient d'ordre 0, $\bar{w}_i$ sont les coefficients d'ordre 1, et $\bar{w}_{ij}$ sont les coefficients d'ordre 2. On peut donc penser qu'intuitivement, plus la valeur de l'ordre d sera élevée et plus l'approximation $\tilde{f}$ de la fonction f sera précise.

Nous pouvons maintenant remarquer que trouver une estimation des coefficients de l'équation (4.9) est un problème standard de régression linéaire où les prédicteurs sont les valeurs des fonctions de Walsh. Les travaux de Leprêtre et al. [57, 56] conseillent d'utiliser l'algorithme Lasso [97] pour apprendre le modèle. En effet, les techniques *"sparse"* peuvent être utilisées pour minimiser le nombre de coefficients non-nuls quand le nombre de prédicteurs est élevé. En outre, les travaux récents [57] en optimisation combinatoire mono-objectif coûteuse considérant des méta-modèles basés sur les fonctions de Walsh ont montré la supériorité de ce méta-modèle sur les méta-modèles présentés précédemment.

Dans nos travaux, qui concernent l'optimisation multi-objectif, nous allons donc nous baser sur la décomposition de Walsh, bien que d'autres modèles de la littérature pourraient être considérés. Le reste de ce chapitre est dédié à la description de notre contribution consistant à proposer et à étudier différents composants nécessaires à la conception d'un framework pour l'optimisation multi-objectif coûteuse de fonctions pseudo-booléennes, ceci en se basant sur le framework MOEA/D.

4.2 Un framework pour l'optimisation multi-objectif combinatoire coûteuse par décomposition

Dans cette section, nous proposons une description détaillée de notre framework S-MCO (*Surrogate-assisted Multiobjective Combinatorial Optimization*), s'inspirant du framework MOEA/D dans le cadre de l'optimisation coûteuse. Comme mentionné précédemment, un composant essentiel de notre framework est l'utilisation du méta-modèle de Walsh, ce qui va naturellement nous conduire à se pencher de façon plus précise sur la construction et l'exploitation de ce type de modèle. En outre, d'autres composants tout aussi essentiels viennent s'ajouter à notre framework pour répondre à des questions qui vont au-delà de notre proposition, à savoir :

— Comment parcourir efficacement l'espace de recherche à l'aide d'un méta-modèle ?
— Comment sélectionner les solutions à évaluer par les fonctions coûteuses ?
— Comment configurer le méta-modèle pour obtenir les meilleurs résultats possibles ?

Ces questions sont traitées de façon incrémentale dans les sections et paragraphes suivants, en commençant d'abord par donner un aperçu général de notre framework, et ensuite en décrivant en détail ses différents composants.

4.2.1 Aperçu général du framework S-MCO

4.2.1.1 Initialisation

Notre framework S-MCO est décrit dans le pseudo-code de haut niveau de l'Algorithme 8. Le framework S-MCO suit dans les grandes lignes le processus de MOEA/D comme nous l'avons décrit dans le Chapitre 1, Section 1.3. Le problème multi-objectif est décomposé en μ sous-problèmes mono-objectifs. La population $\mathcal{P}$ est initialisée avec μ solutions générées aléatoirement. Chacune de ces solutions est évaluée en utilisant les fonctions coûteuses, avant d'être affectée à un sous-problème. Les solutions présentes dans la population initiale sont également utilisées comme première base d'apprentissage pour le méta-modèle. La base d'apprentissage sera notée $\mathcal{D}$. L'algorithme évolue ensuite en parcourant de façon itérative les différents sous-problèmes, de la même façon que la stratégie sps_{ALL} du Chapitre 2. Lors de chaque itération, une nouvelle solution est évaluée en utilisant les fonctions coûteuses jusqu'à ce que le budget disponible soit épuisé.

Algorithme 8 : Surrogate-assisted framework for Multiobjective Combinatorial Optimization (S-MCO)

Input : $\mathcal{W}_\mu := \left\{\omega^1, \ldots, \omega^\mu\right\}$: vecteurs de poids ; $g^{\text{chebyshev}}(\cdot \mid \omega, \cdot)$: fonction d'agrégation que l'on souhaite minimiser ; $\mathcal{O}_{max}$: ordre maximale des fonctions de Walsh ;

$\mathcal{P} \leftarrow \left\{x^1, \ldots, x^\mu\right\}$: population initiale de taille μ;

$\mathcal{D} \leftarrow \left\{(x^1, F(x^1)), \ldots, (x^\mu, F(x^\mu))\right\}$: base d'apprentissage créée à partir de la population initiale ;

$\mathcal{EP} \leftarrow$ initialisation de l'archive externe (facultatif) ;

$z^\star \leftarrow$ initialisation du point de référence avec les solutions présentes dans $\mathcal{P}$;

tant que *le budget global n'est pas épuisé* **faire**

  pour $i \in \{1, \ldots, \mu\}$ **faire**

    /* Choix de l'ordre de Walsh utilisé pendant la prochaine itération i */

    $\mathcal{O} \leftarrow$ ChoixOrdreWalsh($Historique, \mathcal{O}_{max}$);

    /* Entraînement du méta-modèle avec la base d'apprentissage $\mathcal{D}$ et l'ordre de Walsh $\mathcal{O}$ */

    $\widetilde{\mathcal{F}} := (\widetilde{f_1}, \ldots, \widetilde{f_m}) \leftarrow$ Entrainement($\mathcal{D}, \mathcal{O}$);

    /* Création d'une copie du point de référence qui sera utilisé par l'optimiseur et la stratégie de sélection */

    $z^{\star\star} \leftarrow z^\star$;

    /* Exécution de l'optimiseur en utilisant le méta-modèle $\widetilde{\mathcal{F}}$, la population $\mathcal{P}$ et la copie du point de référence $z^{\star\star}$ */

    $\mathcal{S} \leftarrow$ Optimiseur($\mathcal{P}, \widetilde{\mathcal{F}}, z^{\star\star}$) ;

    /* Sélection d'une solution pour l'évaluer avec la vraie fonction coûteuse */

    $x' \leftarrow$ SelectionPourEvaluation($\mathcal{S}, \omega_i, \widetilde{\mathcal{F}}, z^{\star\star}$) ;

    $F(x') \leftarrow$ évaluation de la solution x' ;

    $\mathcal{EP} \leftarrow$ (facultatif) modification de la population externe ;

    $z^\star \leftarrow$ mise à jour du point de référence avec $F(x')$;

    /* Processus de remplacement dans la population $\mathcal{P}$ */

    pour $j \in \{1, \ldots, \mu\}$ **faire**

      si $g^{chebyshev}(x' \mid \omega^j, F)$ ***est meilleur que*** $g^{chebyshev}(x^j \mid \omega^j, F)$ **alors**

        $x^j \leftarrow x'$;

    /* Ajout de la nouvelle solution évaluée dans la base d'apprentissage */

    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x', F(x'))\}$;

4.2.1.2 Construction du méta-modèle

Pour une itération donnée traitant d'un sous-problème $i \in [\![1, \mu]\!]$, la première étape consiste à construire le méta-modèle en utilisant une valeur d'ordre pour les fonctions de Walsh à utiliser. Cette valeur d'ordre est donnée par un composant dédié à chaque itération (ligne 7). Ce composant sera détaillé par la suite dans la Section 4.2.4. On peut noter que ce composant prend en paramètre une variable appelé *Historique*. Cette variable peut être considérée comme un artefact qui indique que ce composant peut utiliser des informations issues de la recherche pour choisir l'ordre de Walsh. Pour chaque objectif f_i de notre problème dans $F = (f_1, \ldots, f_i, \ldots, f_m)$, nous considérons un méta-modèle $\widetilde{f_i}$, pour ainsi obtenir un total de m méta-modèles.

Ces méta-modèles $\widetilde{\mathcal{F}} := (\widetilde{f_1}, \ldots, \widetilde{f_m})$ sont entraînés indépendamment (ligne 8) avec le même ordre de Walsh $\mathcal{O}$ et avec toutes les solutions qui ont été évaluées depuis le début de l'algorithme et qui sont stockées dans la base d'apprentissage $\mathcal{D}$. L'estimation des coefficients de Walsh (équation 4.9) de chacun des méta-modèles $\widetilde{f_i}$ est calculée en suivant la même méthodologie de régression linéaire sparse décrite auparavant dans la Section 4.1.2.4.

4.2.1.3 Exploration de l'espace de recherche

Dans MOEA/D, la population évolue grâce aux nouvelles solutions générées avec des opérateurs de variation comme des mutations ou des croisements. En utilisant ce processus évolutionnaire d'exploration de l'espace de recherche, nous n'avons aucune information sur la qualité des solutions générées avant leur réelle évaluation. Ceci est clairement inacceptable dans le contexte de problèmes coûteux, puisque des solutions de mauvaise qualité pourraient être générées si l'on se basait uniquement sur des opérateurs évolutionnaires classiques. L'idée principale est alors d'assister ce processus d'exploration évolutionnaire de l'espace de recherche à l'aide du méta-modèle construit précédemment. Ainsi, on utilise *un optimiseur* pour explorer intensivement l'espace de recherche sans appeler les fonctions objectif coûteuses.

Dans notre framework, *un optimiseur* (ligne 10) est en réalité un algorithme évolutionnaire faisant appel aux méta-modèles construits, et non aux fonctions coûteuses. Il s'agit ici d'un autre composant, qui prend en paramètre : (1) la population courante $\mathcal{P}$, qui peut notamment être utilisée pour l'étape d'initialisation interne de l'optimiseur, (2) le méta-modèle $\widetilde{\mathcal{F}}$ et (3) une copie du point de référence $z^{\star\star}$ (ligne 9). En sortie, l'optimiseur retourne un ensemble de solutions potentielles, qui représentent au mieux le front Pareto du problème multi-objectif en entrée $\widetilde{\mathcal{F}}$. Il est cependant à noter que

cet ensemble de solution n'a jamais été évalué à l'aide de la fonction coûteuse, mais seulement avec le modèle $\widetilde{\mathcal{F}}$.

Notons à ce stade de la discussion qu'il est important pour l'optimiseur d'utiliser (et de modifier) une copie du point de référence et non le point lui-même $z^\star$. En effet, les solutions qui sont retournées par l'optimiseur ne sont pas évaluées par les vraies fonctions coûteuses. De ce fait, leurs coordonnées dans l'espace objectif ne représentent qu'une approximation de la réalité. On peut alors définir le point $z^{\star\star}$ comme le point représentant les coordonnées de la solution idéale selon l'optimiseur et le méta-modèle à un moment précis du processus de recherche. Ce composant sera étudié plus en détail dans la Section 4.2.2.

4.2.1.4 Sélection de la solution à évaluer

L'optimiseur utilisé dans notre framework retourne un ensemble de solutions de bonne qualité selon le méta-modèle considéré. L'étape suivante consiste à choisir parmi cet ensemble, la ou les solutions qui devront être évaluées par la fonction coûteuse. L'idée est bien sûr de choisir la solution la plus bénéfique à la recherche à l'itération courante i pour améliorer au mieux la population $\mathcal{P}$. Ce composant est appelé *la stratégie de sélection* (ligne 11) dans notre framework S-MCO. Ce composant sera décrit plus en détail dans la Section 4.2.3. Pour l'instant, nous remarquons qu'il prend comme paramètre l'ensemble de solution $\mathcal{S}$ qui ont été générées par l'optimiseur (ligne 10) ainsi que les trois paramètres requis par la fonction d'agrégation pour comparer les solutions, c'est-à-dire le vecteur de poids actuellement parcouru ω_i, le méta-modèle $\widetilde{\mathcal{F}}$ pour avoir une approximation des objectifs, ainsi que la copie du point de référence $z^{\star\star}$ qui a été potentiellement modifiée pendant l'exécution de l'optimiseur. Dans notre framework, la stratégie de sélection retourne une *seule* solution parmi l'ensemble de solutions proposées par l'optimiseur.

4.2.1.5 Mise à jour de la population et de la base d'apprentissage

Une fois la solution candidate identifiée par la stratégie de sélection, la solution va pouvoir être évaluée (ligne 12) par les vraies fonctions coûteuses pour déterminer la qualité réelle de la solution.

Le processus standard de MOEA/D reprend alors le relais en ajoutant la solution x', nouvellement évaluée dans la population externe (ligne 13) si elle ne se fait pas dominer par les solutions déjà présentes. La solution x' y remplacera également les solutions qu'elle domine. Le point de référence $z^\star$ est ensuite mis à jour par les coordonnées réelles

de la solution x' si ses coordonnées dans l'espace objectif améliorent les coordonnées de la solution idéale (ligne 14). On peut noter que la copie du point de référence $z^{\star\star}$ est supprimée à la fin de l'itération pour être reconstruite à la prochaine itération. La population $\mathcal{P}$ est, elle aussi, mise à jour (ligne 15) si la solution x' est meilleure qu'au moins une solution x^i présente dans la population $\mathcal{P}$. La fonction d'agrégation Chebyshev est utilisée ici pour déterminer si une des solutions x^j avec $j \in \{1, \ldots, \mu\}$ de la population $\mathcal{P}$ peut être améliorée par la solution x' pour le sous-problème j. On peut noter que contrairement à MOEA/D, le remplacement se fait sur la totalité de la population $\mathcal{P}$ et non seulement sur le voisinage du sous-problème i de l'itération courante. La suppression du système de voisinage ici nous permet de converger plus rapidement vers le front Pareto. Étant donné que nous avons très peu de budget dans un contexte coûteux, les inconvénients liés à la non-utilisation d'un voisinage comme la perte de diversité peuvent en effet être ignorés.

Grâce à la solution x' nouvellement évaluée, nous allons également pouvoir mettre à jour la base d'apprentissage (ligne 18). Nous avons décidé ici pour plus de simplicité de mettre à jour la base d'apprentissage à chaque fois qu'une nouvelle solution est disponible. Cette mise à jour permet de raffiner le méta-modèle au fur et à mesure des itérations sur les sous-problèmes, afin d'inférer de façon incrémentale une meilleure approximation des fonctions coûteuses.

4.2.2 Composant #1 : Optimiseurs du méta-modèle

Dans le framework S-MCO, les optimiseurs sont à la recherche de solutions pouvant améliorer la population en n'ayant pas le pouvoir d'utiliser les objectifs réels pour évaluer les solutions rencontrées. En effet, l'optimiseur est assisté par notre méta-modèle qui pourra quant à lui donner une approximation de la qualité de chacune des solutions rencontrées. L'intérêt pour l'optimiseur d'utiliser le méta-modèle $\widetilde{\mathcal{F}}$ à la place des vraies fonctions $\mathcal{F}$ est de pouvoir chercher beaucoup plus rapidement, et donc plus intensément, qu'en utilisant les fonctions coûteuses.

Nous considérons dans nos travaux trois optimiseurs du méta-modèle avec des caractéristiques différentes pour mieux analyser l'impact de ce composant sur les performances du framework S-MCO :

$\mathsf{Optim}_{\mathsf{MOEA/D}}$ est un optimiseur qui exécute la version originale de MOEA/D pour un nombre prédéfini de générations, avec pour but d'identifier une approximation du front Pareto pour le méta-modèle $\widetilde{\mathcal{F}}$. L'optimiseur $\mathsf{Optim}_{\mathsf{MOEA/D}}$ est initialisé avec les mêmes paramètres que S-MCO. Il est également initialisé avec

la population $\mathcal{P}$ du framework S-MCO dans l'état actuel avant l'exécution de l'optimiseur. À la fin de son exécution, l'optimiseur retourne la population externe trouvée par $\mathsf{Optim}_{\mathsf{MOEA/D}}$ grâce au méta-modèle. Cette population contient donc un ensemble de solutions non évaluées, mais dont on connaît une approximation de leur qualité.

$\mathsf{Optim}_{\mathsf{MLS}}$ (Algorithme 9) est un optimiseur qui exécute de multiples recherches locales indépendantes. Plus précisément, en utilisant le concept de décomposition dans l'espace objectif, $\mathsf{Optim}_{\mathsf{MLS}}$ exécute une simple recherche locale mono-objectif pour chaque sous-problème défini dans S-MCO, c'est-à-dire pour chaque sous-problème défini par le vecteur de poids ω^i, $i \in [\![1, \mu]\!]$. Nous utilisons un *hill-climber* standard qui utilise le méta-modèle pour évaluer la qualité des solutions. Chaque recherche locale indépendante (par rapport à chaque sous-problème) est initialisée à partir d'une solution générée aléatoirement. Le meilleur voisin améliorant à une distance de Hamming de 1 est choisi à chaque itération de la recherche locale jusqu'à tomber sur un optimum local. Ainsi, $\mathsf{Optim}_{\mathsf{MLS}}$ s'arrête automatiquement quand toutes les recherches locales ne trouvent plus de solutions améliorantes. L'optimiseur retourne ensuite les μ solutions trouvées par les recherches locales, pour chacun des sous-problèmes.

$\mathsf{Optim}_{\mathsf{PLS}}$ (Algorithme 10) est un optimiseur basé sur la recherche locale Pareto (PLS) [80]. Il permet de maintenir une archive non bornée de solutions non-dominées comme peut le faire MOEA/D avec la population externe. À chaque étape, une solution est sélectionnée aléatoirement dans l'archive. Toutes ses solutions voisines sont parcourues pour mettre à jour l'archive. La solution courante est alors marquée comme *visitée* pour éviter d'explorer de nouveau son voisinage. L'optimiseur $\mathsf{Optim}_{\mathsf{PLS}}$ s'arrête quand toutes les solutions de l'archive sont marquées comme *visitée*. Le contenu de l'archive est alors retourné par $\mathsf{Optim}_{\mathsf{PLS}}$.

4.2.3 Composant #2 : Stratégies de sélection

Une fois que l'optimiseur du méta-modèle a terminé son exploration de l'espace de recherche, le framework S-MCO possède dorénavant un ensemble de solutions non évaluées $\mathcal{S}$ (ligne 10). C'est à ce moment que le second composant important du framework rentre en jeu. Il va en effet falloir sélectionner une solution candidate parmi l'ensemble $\mathcal{S}$, solution qui sera ensuite évaluée par les vraies fonctions coûteuses. Étant donné que nous sommes dans un contexte de décomposition, il nous parait

Algorithme 9 : Multiple Local Search (MLS)

Input : $\mathcal{W}_\mu := \left\{w^1, \ldots, w^\mu\right\}$: ensemble de μ vecteurs de poids;
$\widetilde{\mathcal{F}} := (\widetilde{f_1}, \ldots, \widetilde{f_m})$: méta-modèle qui nous donne une approximation de $\mathcal{F}$;
$\mathrm{g}^{\mathsf{chebyshev}}(\cdot \mid \omega, \cdot)$: fonction d'agrégation à minimiser;
$\mathcal{N} : X \leftarrow 2^X$: relation de voisinnage ;

$\mathcal{S} \leftarrow \emptyset$;
pour $\omega^i \in \mathcal{W}_\mu$ **faire**
  LocalOptimum ← False;
  $x^* \leftarrow$ solution (initiale) aléatoire;
  tant que *! LocalOptimum* **faire**
    $x'^* \leftarrow x^*$;
    pour chaque $x' \in \mathcal{N}(x^*)$ **faire**
      si $g^{chebyshev}(x'|\omega^i, \widetilde{\mathcal{F}}) < g^{chebyshev}(x'^*|\omega^i, \widetilde{\mathcal{F}})$ **alors** $x'^* \leftarrow x'$; ;
    si $g^{chebyshev}(x'^*|\omega^i, \widetilde{\mathcal{F}}) < g^{chebyshev}(x^*|\omega^i, \widetilde{\mathcal{F}})$ **alors**
      $x^* \leftarrow x'^*$;
    sinon
      LocalOptimum ← True ;
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{x^*\}$;
retourner $\mathcal{S}$

important pour le choix de cette solution d'utiliser les informations liées au sous-problème courant i, c'est-à-dire la fonction d'agrégation g et le vecteur de poids ω^i. Dans ce contexte où les solutions n'ont jamais été évaluées, nous nous basons sur l'utilisation de la fonction d'agrégation et l'approximation donnée par le méta-modèle pour évaluer le potentiel relatif de chaque solution, et ainsi pouvoir les comparer les unes aux autres. Nous considérons dans nos travaux quatre stratégies différentes pour choisir la meilleure solution candidate pour le sous-problème i parmi l'ensemble de solutions potentielles $\mathcal{S}$. Ceci est expliqué en détail dans ce qui suit :

Select$_{\mathsf{local}}$. Cette première stratégie sélectionne parmi l'ensemble $\mathcal{S}$ la solution x' avec la meilleure valeur d'agrégation pour le sous-problème actuellement visité i et représenté par son vecteur de poids ω^i. Cette stratégie se présente plus formellement sous la forme suivante :

$$x' = \mathrm{argmin}_{x \in \mathcal{S}}\, \mathrm{g}^{\mathsf{chebyshev}}(x \mid \omega^i, \widetilde{\mathcal{F}})$$

Select$_{\mathsf{global}}$. Cette stratégie se base sur la même idée que Select$_{\mathsf{local}}$ mais avec une portée plus globale. La solution candidate x' est en effet la solution avec la

Algorithme 10 : Pareto Local Search (PLS)

Input : $\widetilde{\mathcal{F}} := (\widetilde{f_1}, \ldots, \widetilde{f_m})$: approximation de la fonction $\mathcal{F}$;
$\mathcal{N} : X \leftarrow 2^X$: relation de voisinage ;

```
x ← solution (initiale) aléatoire ;
S ← {x} ;                          // Archive des solutions non dominées
R ← {x} ;                          // Ensemble des solutions non visitées
tant que R ≠ ∅ faire
    x' ← solution aléatoire dans R;
    pour chaque x'' ∈ N(x') faire
        si x'' n'est pas dominéé par une solution de S alors
            pour x ∈ S faire
                si x est dominée par x'' alors S ← S \ {x};
            S ← S ∪ {x''} ;
            pour x ∈ R faire
                si x est dominée par x'' alors R ← R \ {x};
            R ← R ∪ {x''};
    R ← R \ {x'};
retourner S
```

meilleure valeur d'agrégation, mais cette fois sans se limiter au sous-problème courant. La valeur d'agrégation des solutions présentes dans $\mathcal{S}$ est en effet calculée pour chacun des sous-problèmes disponibles. Cette stratégie s'écrit sous la forme suivante :

$$x' := \text{argmin}_{x \in \mathcal{S}} \min_{1 \leq \ell \leq \mu} \text{g}^{\text{chebyshev}}(x \mid \omega^\ell, \widetilde{\mathcal{F}})$$

Select$_{\text{BI}}$. Cette stratégie a pour but de choisir la solution x' qui améliore le mieux la population courante (BI = Best Improvement). On cherche donc à trouver la solution dont l'amélioration de la valeur d'agrégation est la plus grande, pour n'importe quelle solution de la population et ceci indépendamment du sous-problème actuellement parcouru. Cette stratégie s'écrit sous la forme suivante :

$$x' := \text{argmin}_{x \in \mathcal{S}} \min_{1 \leq \ell \leq \mu} \text{g}^{\text{chebyshev}}(x_\ell \mid \omega^\ell, \widetilde{\mathcal{F}}) - \text{g}^{\text{chebyshev}}(x \mid \omega^\ell, \widetilde{\mathcal{F}})$$

Select$_{\text{BI}_{\text{norm}}}$. Cette stratégie est basée sur la stratégie Select$_{\text{BI}}$. La différence avec la stratégie précédente est que la valeur d'amélioration calculée par la stratégie est normalisée par la valeur d'agrégation de la solution courante du sous-problème ℓ.

Cette stratégie s'écrit sous la forme suivante :

$$x' := \text{argmin}_{x \in \mathcal{S}} \min_{1 \leq \ell \leq \mu} \frac{g^{\text{chebyshev}}(x_\ell \mid \omega^\ell, \widetilde{\mathcal{F}}) - g^{\text{chebyshev}}(x \mid \omega^\ell, \widetilde{\mathcal{F}})}{g^{\text{chebyshev}}(x_\ell \mid \omega^\ell, \widetilde{\mathcal{F}})}$$

Afin de déterminer la qualité des solutions présentes dans $\mathcal{S}$, il est important de noter que les stratégies de sélection, décrites dans les paragraphes précédents, utilise exclusivement le méta-modèle $\tilde{\mathcal{F}}$. Autrement dit, l'appel aux vraies fonctions coûteuses n'est jamais effectué à cette étape, ceci afin de préserver au maximum le budget disponible. L'appel aux fonctions coûteuses ne sera effectué qu'à partir de la ligne 12 pour évaluer la solution choisie par la stratégie de sélection.

4.2.4 Composant #3 : Stratégies de choix pour l'ordre de Walsh

Comme nous avons pu le voir dans la Section 4.1.2.4, le méta-modèle basé sur les fonctions de Walsh requiert un ordre $\mathcal{O}$ qui permet de définir le nombre de coefficients à trouver pour approcher au mieux la fonction coûteuse que l'on veut apprendre (équation 4.9). Cet ordre est le seul hyper-paramètre à configurer avant de commencer l'étape d'apprentissage, la base d'apprentissage étant fixée à l'ensemble des solutions déjà évaluées. Cet hyper-paramètre est très important car il définit la précision théorique du modèle de Walsh, ce qui impacte de fait les performances des deux composants présentés précédemment. Le choix de l'ordre de Walsh est une tâche difficile, non seulement car il définit la précision théorique du modèle, mais puisqu'il a également un impact sur le processus d'apprentissage en lui-même. En effet, un ordre de Walsh plus grand permet d'avoir un modèle plus complexe avec une meilleure précision. Encore faut-il être capable d'apprendre correctement les différents coefficients, dont le nombre augmente aussi rapidement. Illustrons ceci sur un problème de taille $n = 50$. Dans ce cas, nous avons potentiellement 51 coefficients pour un ordre $\mathcal{O} = 1$, 1 276 coefficients pour un ordre $\mathcal{O} = 2$ et 20 876 coefficients pour un ordre $\mathcal{O} = 3$. Un nombre élevé de coefficients peut éventuellement améliorer l'approximation de la fonction à apprendre, mais demandera alors une plus grande base d'apprentissage pour entraîner précisément le modèle (sous l'hypothèse d'avoir un algorithme d'apprentissage performant). D'un côté, la base d'apprentissage doit être composée de solutions déjà évaluées. Par conséquence, avoir une grande base d'apprentissage peut être un problème étant donné que nous voulons par définition limiter le nombre de solutions évaluées avec les vraies fonctions coûteuses. D'un autre côté, un faible nombre de coefficients peut être insuffisant pour trouver une approximation précise des fonctions coûteuses.

Nous considérons pour la suite que l'ordre utilisé pour entraîner le méta-modèle de Walsh est limité par une constante $\mathcal{O}_{max}$. Le fait de limiter l'ordre maximal est un choix de conception raisonnable en pratique. Dans le cas contraire, le nombre de coefficients à estimer augmenterait de manière exponentielle, ce qui entraînerait un coût d'apprentissage qui ne serait pas raisonnable et qui pourrait même être supérieur au coût des vraies fonctions $\mathcal{F}$. En outre, de nombreux problèmes d'optimisation combinatoire difficiles peuvent être décomposés de manière exacte en utilisant des fonctions de Walsh avec un ordre borné. Cependant, il demeure difficile d'identifier quel ordre devrait être choisi entre $[\![1, \mathcal{O}_{\max}]\!]$, et de la même manière, d'identifier s'il faut choisir un ordre fixe ou s'il faut mieux faire varier l'ordre pendant le processus de recherche quand le méta-modèle de Walsh est intégré à un algorithme assisté par méta-modèle. Pour essayer de répondre à ces questions, nous proposons d'étudier trois approches différentes pour sélectionner l'ordre de Walsh.

Order$_{\text{static}}$. Avec cette approche, l'ordre de Walsh est un simple paramètre du framework S-MCO qui est paramétré par l'utilisateur avant son exécution. Avec cette stratégie, l'ordre utilisé est fixe et ne change pas au cours du processus de recherche. Le choix de l'ordre par l'utilisateur est très difficile dans un contexte boite-noire où on ne connaît pas les caractéristiques du problème, et donc le nombre de coefficients nécessaires pour trouver une bonne approximation des fonctions coûteuses.

Order$_{\text{random}}$. Dans cette approche, nous proposons de changer l'ordre de Walsh avant chaque nouvel entraînement du modèle de façon aléatoire, uniformément dans l'intervalle $[\![1, \mathcal{O}_{\max}]\!]$. L'avantage de cette approche est que chacun des ordres réalisables sera choisi sans être limité par un ordre fixe qui peut être sous-optimal par rapport à la précision du modèle. Par ailleurs, cette stratégie aléatoire servira de stratégie de référence pour mesurer l'impact de ce composant dans notre analyse expérimentale.

Order$_{\text{greedy}}$. Dans cette approche, nous proposons d'ajuster automatiquement et dynamiquement l'ordre de Walsh à chaque itération, en fonction des données collectées depuis le début du processus de recherche. Plus précisément, à la fin de chaque itération $i \in [\![1, \mu]\!]$, on sauvegarde le nombre de sous-problèmes améliorés par une solution candidate x' qui a été évaluée par les fonctions coûteuses. On notera p_j le nombre de sous-problèmes améliorés, où j correspond à la j^{ieme} évaluation de la fonction coûteuse $\mathcal{F}$ du problème multi-objectif. Ce nombre est ensuite examiné sur une fenêtre glissante des t dernières itérations, où t est un

paramètre défini par l'utilisateur. Au début de chaque nouvelle itération, nous calculons un nouvel indicateur noté $\overline{p}_t$ qui correspond à la moyenne d'amélioration des sous-problèmes sur les t dernières itérations. Si $\overline{p}_t < 1$, autrement dit si nous avons moins d'une amélioration sur les t dernières itérations, alors l'ordre de Walsh est modifié en suivant un processus particulier expliqué ci-dessous.

Définissons $\mathcal{O}_{\text{curr}}$ comme étant l'ordre de Walsh actuellement utilisé par S-MCO à une itération $i \in [\![1, \mu]\!]$. De la même manière, $\mathcal{O}_{\text{prev}}$ est l'ordre de Walsh utilisé à l'itération précédente, autrement dit à l'itération $i-1$. Maintenant, imaginons que nous avons $\overline{p}_t < 1$ ce qui va activer le changement de l'ordre de Walsh. Le nouvel ordre de Walsh que l'on note $\mathcal{O}_{\text{new}}$ est donnée par $\mathcal{O}_{\text{new}} = \mathcal{O}_{\text{curr}} + 1$ si l'ordre courant $\mathcal{O}_{\text{curr}} = 1$ ou si l'ordre précédent $\mathcal{O}_{\text{prev}} < \mathcal{O}_{\text{curr}} < \mathcal{O}_{\text{max}}$. Sinon, le nouvel ordre est donné par $\mathcal{O}_{\text{new}} = \mathcal{O}_{\text{curr}} - 1$.

L'idée derrière cette stratégie dynamique (gloutonne) est de commencer en utilisant le plus petit ordre possible, puis de regarder comment se comporte le processus de recherche en observant la fréquence d'amélioration des différents sous-problèmes. Le même ordre est alors conservé s'il arrive à obtenir au moins une amélioration sur les t dernières itérations. Dans le cas contraire, nous incrémentons l'ordre de Walsh, et de ce fait la complexité du modèle, dans l'espoir d'obtenir un modèle plus précis qui permettra d'identifier des solutions pouvant améliorer notre population. Quand nous atteignons la complexité maximale pour le modèle $\mathcal{O}_{\text{max}}$, nous continuons le processus dans le sens inverse pour revenir vers un modèle plus simple en décrémentant l'ordre de Walsh, qui sera de nouveau incrémenté s'il atteint l'ordre minimal $\mathcal{O} = 1$. Le changement de l'ordre s'arrête quand de nouvelles améliorations sont trouvées, et ce processus est répété jusqu'à l'arrêt de l'algorithme. L'idée est donc de scanner les différents ordres possibles de façon dynamique, en espérant trouver l'ordre le plus adéquat à une étape donnée de la recherche.

4.3 Analyse expérimentale

Le framework S-MCO que nous proposons suit un schéma plug-and-play avec énormément de configurations à étudier. Un défi majeur est d'étudier et d'analyser le plus justement possible l'impact de la combinaison de nos trois principaux composants sur les performances du processus de recherche. Dans la suite, nous commençons par présenter dans la Section 4.3.1 le protocole expérimental suivi à cette fin. Dans la Section 4.3.2, nous présentons notre étude sur l'impact de la stratégie de sélection de

la solution candidate (composant #2) pour chacun des optimiseurs (composant #1) étudiés. Dans la Section 4.3.3, nous montrons les performances relatives des variantes de S-MCO quand la meilleure combinaison des composants #1 et #2 est choisie. Dans la Section 4.3.4, nous étudions l'impact de la configuration de l'ordre de Walsh (composant #3). Nous terminons notre étude avec la Section 4.3.5 en présentant une comparaison des performances des meilleures configurations de S-MCO avec les optimiseurs non-assistés par méta-modèle.

4.3.1 Protocole expérimental

4.3.1.1 Problèmes étudiés

Le but de notre analyse expérimentale est d'étudier les trois principaux composants du framework S-MCO, qui sont l'optimiseur du méta-modèle, la stratégie de sélection de la solution candidate à l'évaluation, et enfin le composant qui choisit l'ordre de Walsh. Nous allons étudier ces différents composants sur des problèmes bi-objectif de type mnk-landscapes et MUBQP (*multi-objective unconstrained binary quadratic programming*). Pour rappel, les problèmes mnk-landscapes nous permettent de faire varier le degré de non-linéarité du problème. Le problème MUBQP est quant à lui un problème difficile qui est connu pour englober une gamme remarquable d'applications en optimisation combinatoire [52]. On considère des instances de mnk-landscapes et MUBQP à deux objectifs ($m = 2$) avec des solutions de taille $n \in \{25, 50\}$. Pour mnk-landscapes, on étudiera des degrés d'épistasie $k \in \{0, 1, 2\}$. Ces problèmes sont en accord avec les études précédentes en optimisation mono-objectif dans un contexte d'optimisation coûteuse [98, 57]. Ce choix nous permet de couvrir des instances allant d'un nombre relativement petit, à relativement grand, de variables, avec un nombre d'interactions allant de linéaire ($k = 0$), à quadratique ($k = 1$) et cubique ($k = 2$).

4.3.1.2 Choix des paramètres

Afin d'examiner de manière approfondie les différents composants introduits dans le framework S-MCO, nous évaluons systématiquement les performances de toutes les combinaisons possibles. Notre but principal est de mener une analyse complète montrant l'impact des différents composants sur le processus de recherche, et de mettre en évidence les problèmes clés pour une recherche efficace de solutions dans un contexte d'optimisation combinatoire multi-objectif assistée par méta-modèle. En plus de cela, étant donné que nous considérons trois optimiseurs du méta-modèle dans notre étude, nous analyserons les performances obtenues avec chaque optimiseur lorsqu'il est utilisé

Tableau 4.1 – Composants et valeurs des configurations du frameworkS-MCO étudiés dans notre analyse expérimentale.

composants	**valeurs**
Optimiseur	$\{\mathsf{Optim}_{\mathsf{MOEA/D}}, \mathsf{Optim}_{\mathsf{MLS}}, \mathsf{Optim}_{\mathsf{PLS}}\}$
Stratégie de sélection	$\left\{\mathsf{Select}_{\mathsf{local}}, \mathsf{Select}_{\mathsf{global}}, \mathsf{Select}_{\mathsf{BI}}, \mathsf{Select}_{\mathsf{BI}_{\mathsf{norm}}}\right\}$
Ordre de Walsh	$\left\{\mathsf{Order}_{\mathsf{static}}, \mathsf{Order}_{\mathsf{random}}, \mathsf{Order}_{\mathsf{greedy}}\right\}$ avec $\mathcal{O} \in \{1, 2, 3\}$

comme algorithme d'optimisation principal pour le problème coûteux, indépendamment du framework S-MCO. Cela nous permet d'analyser les avantages quant à l'utilisation d'une approche assistée par méta-modèle.

Pour toutes les configurations considérées dans cette analyse, nous utilisons 50 vecteurs de poids défini par $\omega^i = \left(\frac{i-1}{\mu-1}, \frac{1-(i-1)}{\mu-1}\right)$, $i \in [\![1, 50]\!]$. Pour les méta-modèles basés sur les fonctions de Walsh, nous utilisons une régression de type Lasso [97] pour estimer les coefficients du modèle. La valeur maximale de l'ordre de Walsh est $\mathcal{O}_{max} = 3$, ce qui correspond à un modèle cubique. Pour rappel, quand nous analysons les variantes de S-MCO avec une stratégie $\mathsf{Order}_{\mathsf{static}}$, l'ordre de Walsh utilisé pour l'entraînement du modèle est un paramètre à définir par l'utilisateur avec $\mathcal{O} \in \{1, 2, 3\}$. La stratégie $\mathsf{Order}_{\mathsf{greedy}}$ adapte l'ordre de Walsh automatiquement en utilisant une fenêtre d'historique de taille $t = 5$. Chaque optimiseur du méta-modèle dépend de ses propres paramètres. $\mathsf{Optim}_{\mathsf{MOEA/D}}$ est exécuté pour 10 générations où il génère des solutions grâce à un croisement à un point suivi d'une mutation avec un taux de $1/n$. La relation de voisinage des optimiseurs $\mathsf{Optim}_{\mathsf{MLS}}$ et $\mathsf{Optim}_{\mathsf{PLS}}$ est basée sur un opérateur bit-flip, c'est-à-dire que deux solutions sont considérées comme voisines si leur distance de Hamming est égale à 1. L'optimiseur $\mathsf{Optim}_{\mathsf{MLS}}$ utilise les mêmes vecteurs de poids que ceux de S-MCO.

4.3.1.3 Évaluation des performances

Globalement, nous expérimentons 60 configurations du framework S-MCO qui sont résumées dans le Tableau 4.1 : 3 optimiseurs × 4 stratégies de sélection × (3 + 2) paramètres liés à l'ordre de Walsh, ainsi que les trois optimiseurs exécutés indépendamment de S-MCO sans méta-modèle. Chaque configuration est exécutée 10 fois indépendamment sur les 8 instances étudiées (2×3 mnk-landscapes + 2 MUBQP), pour un budget maximal de 1 500 évaluations (coûteuses), sans dépasser 64 heures CPU par exécution. Les expériences ont été réalisées avec Python 3.7 sur un processeur Intel Core Xeon

Tableau 4.2 – Rang et moyenne des valeurs de l'indicateur epsilon ($\times 10^2$, entre parenthèse) pour $\text{Optim}_{\text{MOEA/D}}$ après 200, 700, et 1 500 évaluations pour mnk-landscapes.

				$\text{Select}_{\text{local}}$			$\text{Select}_{\text{global}}$			$\text{Select}_{\text{BI}}$			$\text{Select}_{\text{BI}_{\text{norm}}}$		
			Éval.	$\mathcal{O}$=1	$\mathcal{O}$=2	$\mathcal{O}$=3	$\mathcal{O}$=1	$\mathcal{O}$=2	$\mathcal{O}$=3	$\mathcal{O}$=1	$\mathcal{O}$=2	$\mathcal{O}$=3	$\mathcal{O}$=1	$\mathcal{O}$=2	$\mathcal{O}$=3
$\text{Optim}_{\text{MOEA/D}}$	$k=0$	$n=25$	200	$\mathbf{0}_{(1.4)}$	$\mathbf{0}_{(1.8)}$	$3_{(2.4)}$	$9_{(5)}$	$9_{(5.4)}$	$9_{(5.7)}$	$\mathbf{0}_{(1)}$	$\mathbf{0}_{(1.2)}$	$\mathbf{0}_{(1.8)}$	$\mathbf{0}_{(1.3)}$	$\mathbf{0}_{(1.4)}$	$3_{(2.4)}$
			700	$\mathbf{0}_{(1)}$	$\mathbf{0}_{(1.1)}$	$2_{(1.3)}$	$9_{(4.4)}$	$9_{(4.1)}$	$9_{(4.1)}$	$\mathbf{0}_{(0.7)}$	$\mathbf{0}_{(0.7)}$	$\mathbf{0}_{(0.9)}$	$\mathbf{0}_{(1.1)}$	$\mathbf{0}_{(1)}$	$\mathbf{0}_{(1.1)}$
			1 500	$\mathbf{0}_{(0.8)}$	$\mathbf{0}_{(0.8)}$	$\mathbf{0}_{(0.9)}$	$9_{(3.9)}$	$9_{(3.6)}$	$9_{(3.2)}$	$\mathbf{0}_{(0.6)}$	$\mathbf{0}_{(0.6)}$	$\mathbf{0}_{(0.7)}$	$\mathbf{0}_{(0.8)}$	$\mathbf{0}_{(0.9)}$	$\mathbf{0}_{(0.9)}$
		$n=50$	200	$\mathbf{0}_{(2)}$	$2_{(3.7)}$	$2_{(5)}$	$2_{(7.9)}$	$6_{(8.6)}$	$6_{(9.2)}$	$\mathbf{0}_{(2.3)}$	$1_{(3.6)}$	$2_{(4.7)}$	$6_{(8.6)}$	$6_{(9.1)}$	$6_{(9.1)}$
			700	$\mathbf{0}_{(1.4)}$	$\mathbf{0}_{(1.8)}$	$1_{(2.3)}$	$5_{(6.5)}$	$6_{(6.3)}$	$6_{(8)}$	$\mathbf{0}_{(2)}$	$1_{(2.3)}$	$4_{(3)}$	$6_{(8.3)}$	$6_{(8.4)}$	$6_{(8.5)}$
			1 500	$\mathbf{0}_{(1.1)}$	$\mathbf{0}_{(1.3)}$	$\mathbf{0}_{(1.4)}$	$6_{(3.9)}$	$5_{(4.2)}$	$6_{(4.9)}$	$\mathbf{0}_{(1.7)}$	$1_{(1.9)}$	$3_{(2.4)}$	$8_{(7.6)}$	$8_{(7.6)}$	$8_{(7.7)}$
			avg.	**0.56**			7.17			0.67			3.5		
	$k=1$	$n=25$	200	$3_{(4.2)}$	$\mathbf{0}_{(1.5)}$	$\mathbf{0}_{(2.6)}$	$8_{(8.7)}$	$6_{(6.8)}$	$8_{(7.7)}$	$3_{(4.5)}$	$\mathbf{0}_{(1.1)}$	$\mathbf{0}_{(2.5)}$	$3_{(4.2)}$	$8_{(7.2)}$	$\mathbf{0}_{(1.7)}$
			700	$5_{(3.4)}$	$\mathbf{0}_{(0.5)}$	$\mathbf{0}_{(0.5)}$	$7_{(6.6)}$	$6_{(5.7)}$	$7_{(6.2)}$	$5_{(3.8)}$	$\mathbf{0}_{(0.5)}$	$\mathbf{0}_{(0.6)}$	$5_{(2.8)}$	$7_{(6)}$	$\mathbf{0}_{(0.6)}$
			1 500	$5_{(2.9)}$	$\mathbf{0}_{(0.3)}$	$\mathbf{0}_{(0.4)}$	$7_{(5.7)}$	$5_{(4.4)}$	$6_{(4.8)}$	$5_{(3.5)}$	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(0.4)}$	$5_{(2.6)}$	$7_{(5.7)}$	$\mathbf{0}_{(0.4)}$
		$n=50$	200	$\mathbf{0}_{(5.2)}$	$\mathbf{0}_{(4.8)}$	$1_{(6.3)}$	$6_{(9.7)}$	$6_{(10.6)}$	$6_{(10.9)}$	$\mathbf{0}_{(5.8)}$	$\mathbf{0}_{(4.8)}$	$\mathbf{0}_{(5.6)}$	$6_{(9.8)}$	$6_{(10.7)}$	$6_{(10.9)}$
			700	$1_{(4.1)}$	$\mathbf{0}_{(3.2)}$	$\mathbf{0}_{(3.7)}$	$6_{(7.7)}$	$5_{(7.7)}$	$6_{(9.1)}$	$1_{(4.4)}$	$\mathbf{0}_{(2.8)}$	$\mathbf{0}_{(3.4)}$	$6_{(8.7)}$	$6_{(9.6)}$	$7_{(9.9)}$
			1 500	$1_{(3.8)}$	$\mathbf{0}_{(2.9)}$	$\mathbf{0}_{(3.1)}$	$4_{(6.5)}$	$\mathbf{0}_{(5.6)}$	$3_{(6)}$	$\mathbf{0}_{(3.9)}$	$\mathbf{0}_{(2.8)}$	$\mathbf{0}_{(2.8)}$	$6_{(8.3)}$	$6_{(8.6)}$	$6_{(8.7)}$
			avg.	0.89			5.67			**0.78**			5		
	$k=2$	$n=25$	200	$4_{(8)}$	$\mathbf{0}_{(5.6)}$	$\mathbf{0}_{(5.9)}$	$6_{(9)}$	$6_{(8.3)}$	$4_{(8.4)}$	$2_{(7.5)}$	$\mathbf{0}_{(4.3)}$	$\mathbf{0}_{(4.7)}$	$5_{(8.2)}$	$\mathbf{0}_{(4.7)}$	$\mathbf{0}_{(5.2)}$
			700	$6_{(6.5)}$	$\mathbf{0}_{(3.9)}$	$\mathbf{0}_{(3)}$	$6_{(7.7)}$	$1_{(5.4)}$	$\mathbf{0}_{(6.2)}$	$6_{(7.2)}$	$\mathbf{0}_{(2.9)}$	$\mathbf{0}_{(2.7)}$	$5_{(6.8)}$	$\mathbf{0}_{(3.6)}$	$\mathbf{0}_{(3.2)}$
			1 500	$6_{(6.2)}$	$\mathbf{0}_{(3.4)}$	$\mathbf{0}_{(2)}$	$6_{(7.2)}$	$\mathbf{0}_{(5)}$	$\mathbf{0}_{(5.3)}$	$5_{(5.9)}$	$\mathbf{0}_{(2.4)}$	$\mathbf{0}_{(2.1)}$	$5_{(6.1)}$	$\mathbf{0}_{(2.8)}$	$\mathbf{0}_{(2.7)}$
		$n=50$	200	$\mathbf{0}_{(8)}$	$\mathbf{0}_{(7.7)}$	$\mathbf{0}_{(7.7)}$	$5_{(13.3)}$	$5_{(11.6)}$	$5_{(12.3)}$	$\mathbf{0}_{(9.4)}$	$\mathbf{0}_{(6.5)}$	$\mathbf{0}_{(7.3)}$	$5_{(13.3)}$	$5_{(11.6)}$	$5_{(12.3)}$
			700	$\mathbf{0}_{(5.7)}$	$\mathbf{0}_{(4.4)}$	$\mathbf{0}_{(4.6)}$	$6_{(11.2)}$	$5_{(9.6)}$	$6_{(10.6)}$	$\mathbf{0}_{(6.3)}$	$\mathbf{0}_{(4.6)}$	$\mathbf{0}_{(4.4)}$	$6_{(11.4)}$	$6_{(10.6)}$	$6_{(11.8)}$
			1 500	$1_{(5.4)}$	$\mathbf{0}_{(3.4)}$	$\mathbf{0}_{(3.3)}$	$6_{(10.7)}$	$4_{(7.8)}$	$4_{(8.3)}$	$\mathbf{0}_{(5.3)}$	$\mathbf{0}_{(4.1)}$	$\mathbf{0}_{(3.7)}$	$6_{(10.9)}$	$6_{(9.6)}$	$6_{(10.5)}$
			avg.	0.94			4.17			**0.72**			3.67		

E5-2630 (2.20 GHz, 256 Go RAM) qui fonctionne sous Debian 10. On peut noter que nous présentons la qualité de chaque algorithme pour différents budgets intermédiaires, ce qui va nous permettre d'étudier le profil de convergence de chaque configuration possible de S-MCO.

Pour l'évaluation des performances, nous utilisons l'indicateur de performance epsilon, décrit dans le Chapitre 1. On peut noter cependant que nos résultats sont similaires avec l'indicateur d'hypervolume, que nous n'afficherons pas ici pour plus de clarté dans la présentation des résultats. Pour une exécution donnée, l'archive externe contenant toutes les solutions non-dominées rencontrées durant le processus de recherche est utilisée pour calculer la qualité de l'approximation du front Pareto.

4.3.2 Analyse de l'impact des stratégies de sélection sur les optimiseurs du méta-modèle

Nous commençons par étudier l'impact des différentes stratégies de sélection (composant #2) sur chaque variante de S-MCO utilisant un optimiseur du méta-modèle différent (composant #1). Les résultats sont résumés dans les Tableaux 4.2, 4.3, 4.4, et 4.5. Chacun de ces tableaux représente les rangs d'une configuration donnée pour

Tableau 4.3 – Rang et moyenne des valeurs de l'indicateur epsilon ($\times 10^2$, entre parenthèse) pour $\text{Optim}_{\text{MLS}}$ après 200, 700, et 1 500 évaluations pour mnk-landscapes.

				$\text{Select}_{\text{local}}$			$\text{Select}_{\text{global}}$			$\text{Select}_{\text{BI}}$			$\text{Select}_{\text{BI}_{\text{norm}}}$		
			Éval.	$\mathcal{O}$=1	$\mathcal{O}$=2	$\mathcal{O}$=3	$\mathcal{O}$=1	$\mathcal{O}$=2	$\mathcal{O}$=3	$\mathcal{O}$=1	$\mathcal{O}$=2	$\mathcal{O}$=3	$\mathcal{O}$=1	$\mathcal{O}$=2	$\mathcal{O}$=3
$\text{Optim}_{\text{MLS}}$	$k=0$	$n=25$	200	$5_{(4.7)}$	$3_{(3.3)}$	$2_{(2.4)}$	$3_{(2.2)}$	$4_{(4.7)}$	$7_{(5.2)}$	$5_{(3.7)}$	$3_{(3.2)}$	$3_{(2.8)}$	$\mathbf{0}_{(1.1)}$	$\mathbf{0}_{(\underline{0.4})}$	$1_{(1)}$
			700	$4_{(3.1)}$	$3_{(2.4)}$	$3_{(1.5)}$	$3_{(2.1)}$	$8_{(4.1)}$	$8_{(4.7)}$	$3_{(2.9)}$	$3_{(2.3)}$	$2_{(1.8)}$	$\mathbf{0}_{(0.3)}$	$\mathbf{0}_{(0.3)}$	$\mathbf{0}_{(\underline{0.3})}$
			1 500	$3_{(2.3)}$	$3_{(1.7)}$	$\mathbf{0}_{(1.1)}$	$3_{(2.1)}$	$10_{(3.9)}$	$10_{(4)}$	$3_{(2.2)}$	$3_{(1.7)}$	$2_{(1.3)}$	$\mathbf{0}_{(0.3)}$	$\mathbf{0}_{(0.3)}$	$\mathbf{0}_{(\underline{0.3})}$
		$n=50$	200	$3_{(5.6)}$	$1_{(4.6)}$	$1_{(4.9)}$	$1_{(4.1)}$	$5_{(6.8)}$	$10_{(7.4)}$	$1_{(3.7)}$	$1_{(4.1)}$	$2_{(5.4)}$	$1_{(3.9)}$	$\mathbf{0}_{(\underline{1.7})}$	$1_{(4.5)}$
			700	$9_{(4.3)}$	$1_{(1.2)}$	$3_{(1.2)}$	$4_{(1.6)}$	$10_{(6.2)}$	$10_{(6.4)}$	$5_{(1.9)}$	$\mathbf{0}_{(0.6)}$	$2_{(1)}$	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(\underline{0.3})}$	$\mathbf{0}_{(0.5)}$
			1 500	$9_{(3.3)}$	$1_{(0.5)}$	$\mathbf{0}_{(0.3)}$	$8_{(1.6)}$	$10_{(5.9)}$	$10_{(6.2)}$	$6_{(0.8)}$	$\mathbf{0}_{(0.3)}$	$\mathbf{0}_{(\underline{0.2})}$	$\mathbf{0}_{(0.3)}$	$\mathbf{0}_{(0.3)}$	$\mathbf{0}_{(0.3)}$
			avg.	3			6.89			2.44			**0.17**		
	$k=1$	$n=25$	200	$2_{(3.9)}$	$1_{(2.7)}$	$1_{(2.8)}$	$6_{(6.3)}$	$4_{(5.4)}$	$6_{(6.3)}$	$3_{(4.8)}$	$1_{(1.5)}$	$1_{(3.2)}$	$4_{(4.7)}$	$\mathbf{0}_{(\underline{0.6})}$	$1_{(1.7)}$
			700	$6_{(2.9)}$	$3_{(1.2)}$	$1_{(0.6)}$	$9_{(5.5)}$	$7_{(4.8)}$	$8_{(5.2)}$	$6_{(3.9)}$	$2_{(1)}$	$1_{(0.9)}$	$6_{(3.8)}$	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(\underline{0.3})}$
			1 500	$6_{(2.6)}$	$3_{(0.9)}$	$1_{(0.5)}$	$9_{(5.1)}$	$7_{(4.4)}$	$9_{(4.9)}$	$6_{(3.7)}$	$2_{(0.8)}$	$1_{(0.8)}$	$6_{(3.5)}$	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(\underline{0.3})}$
		$n=50$	200	$1_{(4.7)}$	$2_{(5.9)}$	$1_{(6.2)}$	$1_{(5.2)}$	$5_{(6.7)}$	$8_{(8.2)}$	$\mathbf{0}_{(4.3)}$	$1_{(5.3)}$	$3_{(6.1)}$	$\mathbf{0}_{(4.4)}$	$\mathbf{0}_{(\underline{3.2})}$	$\mathbf{0}_{(4.5)}$
			700	$5_{(3.3)}$	$1_{(1.6)}$	$1_{(2.4)}$	$8_{(5.1)}$	$9_{(6.5)}$	$9_{(6.6)}$	$5_{(3.2)}$	$\mathbf{0}_{(0.8)}$	$2_{(1.4)}$	$5_{(3.5)}$	$\mathbf{0}_{(\underline{0.4})}$	$1_{(1.3)}$
			1 500	$6_{(2.9)}$	$4_{(1)}$	$\mathbf{0}_{(0.6)}$	$9_{(5)}$	$9_{(6.4)}$	$9_{(6.5)}$	$6_{(3.2)}$	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(0.4)}$	$6_{(3.4)}$	$\mathbf{0}_{(\underline{0.3})}$	$\mathbf{0}_{(0.4)}$
			avg.	2.5			7.33			2.22			**1.61**		
	$k=2$	$n=25$	200	$\mathbf{0}_{(7.4)}$	$\mathbf{0}_{(6.4)}$	$\mathbf{0}_{(5.9)}$	$4_{(8.7)}$	$\mathbf{0}_{(\underline{5})}$	$1_{(7.1)}$	$1_{(8.5)}$	$\mathbf{0}_{(5.7)}$	$\mathbf{0}_{(6.1)}$	$1_{(8.3)}$	$\mathbf{0}_{(5.3)}$	$\mathbf{0}_{(5.5)}$
			700	$4_{(5.6)}$	$3_{(2.8)}$	$\mathbf{0}_{(1.1)}$	$8_{(7.4)}$	$3_{(3.6)}$	$4_{(4.8)}$	$7_{(7.1)}$	$2_{(2.9)}$	$\mathbf{0}_{(0.9)}$	$7_{(7.3)}$	$2_{(2.5)}$	$\mathbf{0}_{(\underline{0.5})}$
			1 500	$6_{(5.1)}$	$3_{(2)}$	$\mathbf{0}_{(0.7)}$	$8_{(7.1)}$	$6_{(3.5)}$	$6_{(4.6)}$	$7_{(6.6)}$	$3_{(2.1)}$	$\mathbf{0}_{(0.5)}$	$8_{(6.9)}$	$3_{(2)}$	$\mathbf{0}_{(\underline{0.5})}$
		$n=50$	200	$\mathbf{0}_{(7)}$	$\mathbf{0}_{(7.7)}$	$\mathbf{0}_{(8.4)}$	$8_{(10.7)}$	$\mathbf{0}_{(8.8)}$	$1_{(9.5)}$	$\mathbf{0}_{(8.3)}$	$\mathbf{0}_{(7.3)}$	$\mathbf{0}_{(7.6)}$	$1_{(9.1)}$	$\mathbf{0}_{(7)}$	$\mathbf{0}_{(\underline{6.3})}$
			700	$2_{(4.7)}$	$\mathbf{0}_{(3.6)}$	$\mathbf{0}_{(3.7)}$	$9_{(10)}$	$5_{(7.2)}$	$7_{(8.3)}$	$6_{(6.3)}$	$\mathbf{0}_{(3.2)}$	$\mathbf{0}_{(4)}$	$6_{(6.6)}$	$\mathbf{0}_{(\underline{2.8})}$	$\mathbf{0}_{(3.3)}$
			1 500	$6_{(4.3)}$	$\mathbf{0}_{(1.9)}$	$\mathbf{0}_{(1.9)}$	$11_{(9.8)}$	$6_{(6.7)}$	$6_{(7.3)}$	$7_{(6.1)}$	$\mathbf{0}_{(1.8)}$	$\mathbf{0}_{(2)}$	$7_{(6)}$	$\mathbf{0}_{(\underline{1.4})}$	$\mathbf{0}_{(1.5)}$
			avg.	**1.33**			5.17			1.83			1.94		

Tableau 4.4 – Rang et moyenne des valeurs de l'indicateur epsilon ($\times 10^2$, entre parenthèse) pour $\text{Optim}_{\text{PLS}}$ après 200, 700, et 1 500 évaluations pour mnk-landscapes.

				$\text{Select}_{\text{local}}$			$\text{Select}_{\text{global}}$			$\text{Select}_{\text{BI}}$			$\text{Select}_{\text{BI}_{\text{norm}}}$		
			Éval.	$\mathcal{O}$=1	$\mathcal{O}$=2	$\mathcal{O}$=3	$\mathcal{O}$=1	$\mathcal{O}$=2	$\mathcal{O}$=3	$\mathcal{O}$=1	$\mathcal{O}$=2	$\mathcal{O}$=3	$\mathcal{O}$=1	$\mathcal{O}$=2	$\mathcal{O}$=3
$\text{Optim}_{\text{PLS}}$	$k=0$	$n=25$	200	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(\underline{0.4})}$	$6_{(1.7)}$	$10_{(6.4)}$	$9_{(5.4)}$	$9_{(4.9)}$	$\mathbf{0}_{(0.5)}$	$3_{(0.8)}$	$3_{(1.2)}$	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(0.5)}$	$2_{(1)}$
			700	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(\underline{0.3})}$	$9_{(5.4)}$	$9_{(4.6)}$	$9_{(4.4)}$	$1_{(0.5)}$	$4_{(0.8)}$	$4_{(1.1)}$	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(0.5)}$	$\mathbf{0}_{(0.5)}$
			1 500	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(\underline{0.3})}$	$10_{(4.7)}$	$9_{(4.3)}$	$9_{(3.9)}$	$1_{(0.5)}$	$4_{(0.8)}$	$4_{(1.1)}$	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(0.5)}$	$\mathbf{0}_{(0.5)}$
		$n=50$	200	$\mathbf{0}_{(\underline{0.4})}$	$3_{(2.3)}$	$6_{(4.7)}$	$9_{(8.1)}$	$9_{(6.9)}$	$7_{(6.6)}$	$\mathbf{0}_{(0.5)}$	$3_{(2.2)}$	$6_{(4.7)}$	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(1.5)}$	$4_{(4.1)}$
			700	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(\underline{0.3})}$	$2_{(0.7)}$	$10_{(7.2)}$	$9_{(6.3)}$	$9_{(6)}$	$1_{(0.5)}$	$\mathbf{0}_{(0.6)}$	$4_{(0.8)}$	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(0.3)}$	$1_{(0.5)}$
			1 500	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(0.3)}$	$\mathbf{0}_{(\underline{0.2})}$	$9_{(6.7)}$	$9_{(6.1)}$	$9_{(5.9)}$	$2_{(0.5)}$	$\mathbf{0}_{(0.6)}$	$6_{(0.7)}$	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(0.3)}$	$\mathbf{0}_{(0.4)}$
			avg.	0.94			9.06			2.56			**0.39**		
	$k=1$	$n=25$	200	$5_{(3.7)}$	$\mathbf{0}_{(0.8)}$	$1_{(1.5)}$	$8_{(6.8)}$	$8_{(6.5)}$	$7_{(6.1)}$	$6_{(5)}$	$\mathbf{0}_{(0.9)}$	$1_{(2.1)}$	$5_{(4.4)}$	$\mathbf{0}_{(\underline{0.5})}$	$1_{(1.3)}$
			700	$6_{(2.8)}$	$\mathbf{0}_{(0.3)}$	$\mathbf{0}_{(\underline{0.3})}$	$8_{(5.6)}$	$8_{(5.6)}$	$7_{(5.5)}$	$7_{(4.6)}$	$\mathbf{0}_{(0.6)}$	$2_{(0.6)}$	$6_{(4.1)}$	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(0.4)}$
			1 500	$6_{(2.8)}$	$\mathbf{0}_{(0.3)}$	$\mathbf{0}_{(\underline{0.3})}$	$8_{(5.2)}$	$7_{(4.8)}$	$7_{(5)}$	$7_{(4.3)}$	$\mathbf{0}_{(0.6)}$	$3_{(0.6)}$	$6_{(4)}$	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(0.4)}$
		$n=50$	200	$1_{(4.6)}$	$\mathbf{0}_{(3.5)}$	$2_{(5.4)}$	$7_{(7.1)}$	$8_{(7.5)}$	$7_{(7.7)}$	$1_{(4.9)}$	$1_{(4)}$	$1_{(5.2)}$	$1_{(4.6)}$	$\mathbf{0}_{(\underline{2.4})}$	$1_{(4.3)}$
			700	$6_{(3.5)}$	$\mathbf{0}_{(0.6)}$	$3_{(1.7)}$	$9_{(6.7)}$	$9_{(7.3)}$	$9_{(6.7)}$	$6_{(4.1)}$	$\mathbf{0}_{(0.6)}$	$3_{(1.9)}$	$6_{(4)}$	$\mathbf{0}_{(\underline{0.4})}$	$1_{(1)}$
			1 500	$6_{(3.4)}$	$\mathbf{0}_{(\underline{0.4})}$	$\mathbf{0}_{(0.4)}$	$9_{(6.6)}$	$9_{(7.1)}$	$9_{(6.4)}$	$6_{(4)}$	$\mathbf{0}_{(0.6)}$	$\mathbf{0}_{(0.5)}$	$6_{(4)}$	$\mathbf{0}_{(0.4)}$	$\mathbf{0}_{(0.4)}$
			avg.	2			8			2.44			**1.83**		
	$k=2$	$n=25$	200	$3_{(6.4)}$	$\mathbf{0}_{(4.8)}$	$\mathbf{0}_{(5)}$	$9_{(8.9)}$	$\mathbf{0}_{(5.8)}$	$\mathbf{0}_{(6.4)}$	$3_{(7.7)}$	$\mathbf{0}_{(3.9)}$	$\mathbf{0}_{(5.3)}$	$1_{(6.8)}$	$\mathbf{0}_{(\underline{3.8})}$	$\mathbf{0}_{(4)}$
			700	$7_{(6)}$	$3_{(2.3)}$	$\mathbf{0}_{(1)}$	$8_{(8)}$	$4_{(3.6)}$	$4_{(4.7)}$	$5_{(5.9)}$	$3_{(2.5)}$	$\mathbf{0}_{(0.7)}$	$6_{(5.9)}$	$3_{(2.1)}$	$\mathbf{0}_{(\underline{0.7})}$
			1 500	$6_{(5.7)}$	$3_{(1.8)}$	$\mathbf{0}_{(\underline{0.2})}$	$8_{(7.1)}$	$6_{(3.6)}$	$6_{(4.5)}$	$6_{(5.4)}$	$3_{(2.3)}$	$\mathbf{0}_{(0.6)}$	$7_{(5.7)}$	$3_{(1.9)}$	$\mathbf{0}_{(0.5)}$
		$n=50$	200	$2_{(8)}$	$\mathbf{0}_{(\underline{5.6})}$	$\mathbf{0}_{(6.6)}$	$7_{(10.7)}$	$\mathbf{0}_{(8.1)}$	$5_{(9.5)}$	$5_{(9)}$	$\mathbf{0}_{(6.4)}$	$\mathbf{0}_{(6.7)}$	$4_{(8.5)}$	$\mathbf{0}_{(6.2)}$	$\mathbf{0}_{(5.9)}$
			700	$4_{(5.9)}$	$\mathbf{0}_{(3)}$	$\mathbf{0}_{(3.8)}$	$10_{(9.9)}$	$4_{(6.6)}$	$6_{(7.8)}$	$4_{(6.2)}$	$\mathbf{0}_{(2.7)}$	$\mathbf{0}_{(4)}$	$6_{(6.3)}$	$\mathbf{0}_{(\underline{2.4})}$	$\mathbf{0}_{(3.3)}$
			1 500	$6_{(5.2)}$	$\mathbf{0}_{(2.2)}$	$\mathbf{0}_{(2.3)}$	$11_{(9.3)}$	$6_{(5.6)}$	$7_{(7.2)}$	$6_{(5.7)}$	$\mathbf{0}_{(1.9)}$	$\mathbf{0}_{(2.6)}$	$6_{(6)}$	$\mathbf{0}_{(\underline{1.8})}$	$\mathbf{0}_{(1.8)}$
			avg.	**1.89**			5.61			1.94			2		

Tableau 4.5 – Rang et moyenne des valeurs de l'indicateur epsilon ($\times 10^{-2}$, entre parenthèses) pour les différents algorithmes après 200, 700, et 1 500 évaluations pour les instances de UBQP.

			$\text{Select}_{\text{local}}$			$\text{Select}_{\text{global}}$			$\text{Select}_{\text{BI}}$			$\text{Select}_{\text{BI}_{\text{norm}}}$		
		Eval.	$\mathcal{O}$=1	$\mathcal{O}$=2	$\mathcal{O}$=3	$\mathcal{O}$=1	$\mathcal{O}$=2	$\mathcal{O}$=3	$\mathcal{O}$=1	$\mathcal{O}$=2	$\mathcal{O}$=3	$\mathcal{O}$=1	$\mathcal{O}$=2	$\mathcal{O}$=3
$\text{Optim}_{\text{MOEA/D}}$	n = 25	200	$1_{(3.7)}$	$\mathbf{0}_{(1.9)}$	$\mathbf{0}_{(2.5)}$	$5_{(7.6)}$	$4_{(6.4)}$	$3_{(6.6)}$	$3_{(5.2)}$	$\mathbf{0}_{(\underline{1.8})}$	$\mathbf{0}_{(2.3)}$	$5_{(7.8)}$	$4_{(6.7)}$	$5_{(7.1)}$
		700	$4_{(2.8)}$	$\mathbf{0}_{(\underline{0.3})}$	$1_{(1.1)}$	$6_{(6.4)}$	$4_{(5)}$	$4_{(3.9)}$	$4_{(3.9)}$	$\mathbf{0}_{(0.6)}$	$1_{(1)}$	$5_{(6.6)}$	$4_{(5.7)}$	$4_{(4.8)}$
		1500	$4_{(2.6)}$	$\mathbf{0}_{(\underline{0.2})}$	$\mathbf{0}_{(0.5)}$	$8_{(5.8)}$	$4_{(3.4)}$	$4_{(2.9)}$	$4_{(3.7)}$	$1_{(0.5)}$	$\mathbf{0}_{(0.5)}$	$8_{(5.8)}$	$4_{(5.3)}$	$4_{(2.8)}$
	n = 50	200	$\mathbf{0}_{(\underline{14.2})}$	$\mathbf{0}_{(18.5)}$	$\mathbf{0}_{(18.4)}$	$6_{(33)}$	$6_{(29.6)}$	$6_{(29.9)}$	$\mathbf{0}_{(15.5)}$	$\mathbf{0}_{(17.8)}$	$\mathbf{0}_{(17.3)}$	$6_{(33.2)}$	$6_{(28.6)}$	$6_{(30.3)}$
		700	$\mathbf{0}_{(\underline{9.4})}$	$\mathbf{0}_{(10.6)}$	$\mathbf{0}_{(10.3)}$	$6_{(27.1)}$	$6_{(23.6)}$	$5_{(20.8)}$	$\mathbf{0}_{(9.4)}$	$\mathbf{0}_{(10.5)}$	$\mathbf{0}_{(9.8)}$	$7_{(30)}$	$6_{(25.5)}$	$5_{(22.8)}$
		1500	$1_{(8.3)}$	$\mathbf{0}_{(\underline{5.6})}$	$1_{(8.8)}$	$6_{(22.2)}$	$6_{(16.6)}$	$2_{(14)}$	$1_{(8.3)}$	$\mathbf{0}_{(6.7)}$	$1_{(9.4)}$	$8_{(27.6)}$	$7_{(23.6)}$	$1_{(16.4)}$
		avg.	**0.67**			5.06			0.83			5.28		
$\text{Optim}_{\text{MLS}}$	n = 25	200	$\mathbf{0}_{(2.9)}$	$\mathbf{0}_{(4)}$	$2_{(4.1)}$	$10_{(7.1)}$	$2_{(4.2)}$	$2_{(4.7)}$	$\mathbf{0}_{(3.6)}$	$\mathbf{0}_{(3.3)}$	$\mathbf{0}_{(3.4)}$	$\mathbf{0}_{(4.3)}$	$\mathbf{0}_{(\underline{2.3})}$	$\mathbf{0}_{(3.2)}$
		700	$5_{(2.4)}$	$1_{(0.7)}$	$5_{(2.4)}$	$11_{(6.5)}$	$7_{(4.1)}$	$6_{(3.5)}$	$6_{(3.4)}$	$\mathbf{0}_{(0.5)}$	$1_{(1.1)}$	$5_{(4)}$	$\mathbf{0}_{(\underline{0.2})}$	$1_{(0.6)}$
		1500	$5_{(2.4)}$	$1_{(0.4)}$	$2_{(0.9)}$	$11_{(5.8)}$	$7_{(4.1)}$	$6_{(3.3)}$	$6_{(3.3)}$	$\mathbf{0}_{(0.4)}$	$1_{(0.6)}$	$6_{(3.8)}$	$\mathbf{0}_{(\underline{0.1})}$	$1_{(0.3)}$
	n = 50	200	$\mathbf{0}_{(\underline{11.3})}$	$3_{(28)}$	$3_{(26.7)}$	$3_{(23.5)}$	$3_{(26.6)}$	$3_{(27.8)}$	$\mathbf{0}_{(12.6)}$	$3_{(30.8)}$	$3_{(27.4)}$	$\mathbf{0}_{(12.4)}$	$3_{(26)}$	$3_{(28.2)}$
		700	$\mathbf{0}_{(\underline{6})}$	$3_{(15.6)}$	—	$7_{(23.1)}$	$3_{(15.8)}$	$\mathbf{0}_{(18.4)}$	$1_{(8.6)}$	$3_{(13.7)}$	—	$1_{(8.3)}$	$3_{(16)}$	$\mathbf{0}_{(16.1)}$
		1500	$2_{(5.6)}$	$2_{(7.9)}$	—	$7_{(22.7)}$	$6_{(14)}$	—	$3_{(8.6)}$	$\mathbf{0}_{(1.5)}$	—	$3_{(8.3)}$	$\mathbf{0}_{(\underline{1.2})}$	—
		avg.	1.89			5.22			1.5			**1.44**		
$\text{Optim}_{\text{PLS}}$	n = 25	200	$\mathbf{0}_{(3.4)}$	$\mathbf{0}_{(2.2)}$	$\mathbf{0}_{(2.5)}$	$11_{(7.7)}$	$4_{(5.2)}$	$1_{(4.6)}$	$1_{(4.1)}$	$\mathbf{0}_{(\underline{2})}$	$\mathbf{0}_{(2.7)}$	$2_{(4.4)}$	$\mathbf{0}_{(3.4)}$	$\mathbf{0}_{(3.6)}$
		700	$6_{(2.7)}$	$\mathbf{0}_{(\underline{0.2})}$	$1_{(1.2)}$	$10_{(6.4)}$	$9_{(4.9)}$	$6_{(3.9)}$	$6_{(3.1)}$	$1_{(0.7)}$	$1_{(0.7)}$	$6_{(3.2)}$	$\mathbf{0}_{(0.6)}$	$1_{(1.1)}$
		1500	$6_{(2.6)}$	$\mathbf{0}_{(\underline{0.2})}$	$\mathbf{0}_{(0.5)}$	$11_{(6.1)}$	$9_{(4.7)}$	$6_{(3.7)}$	$6_{(2.9)}$	$1_{(0.7)}$	$1_{(0.6)}$	$6_{(3.1)}$	$\mathbf{0}_{(0.6)}$	$1_{(0.7)}$
	n = 50	200	$\mathbf{0}_{(\underline{11.9})}$	$3_{(21.4)}$	$3_{(21.1)}$	$8_{(29.4)}$	$3_{(24.7)}$	$3_{(24.1)}$	$\mathbf{0}_{(13.1)}$	$3_{(23.5)}$	$3_{(24)}$	$\mathbf{0}_{(13.1)}$	$3_{(24.7)}$	$3_{(24.1)}$
		700	$\mathbf{0}_{(\underline{7.8})}$	$\mathbf{0}_{(10.5)}$	—	$7_{(27.3)}$	$6_{(15.9)}$	—	$1_{(11.2)}$	$\mathbf{0}_{(9)}$	—	$\mathbf{0}_{(10.2)}$	$\mathbf{0}_{(9.5)}$	—
		1500	$3_{(7.3)}$	$\mathbf{0}_{(1.4)}$	—	$7_{(26)}$	$6_{(15.9)}$	—	$3_{(10.5)}$	$\mathbf{0}_{(1.8)}$	—	$3_{(9.6)}$	$\mathbf{0}_{(\underline{1.2})}$	—
		avg.	**1.22**			5.94			1.5			1.39		

chacune des stratégies de sélection étudiées et pour chaque ordre (statique) de Walsh. Les rangs sont calculés en utilisant un test statistique de Wilcoxon avec une valeur de confiance à 0.05, comme expliqué dans les chapitres précédents. En particulier, un rang de 0 indique qu'il n'existe aucune configuration qui domine significativement la configuration courante. On peut noter que l'on considère différents budgets, allant de peu à relativement beaucoup d'évaluations coûteuses, ceci pour permettre l'étude des comportements *anytime* des configurations.

Ces tableaux nous permettent d'identifier deux observations à deux niveaux différents de notre framework S-MCO. La première observation est au niveau des stratégies de sélection. En effet, on remarque que le rang obtenu pour une stratégie de sélection donnée est globalement homogène sur les différentes instances considérées. Ceci bien que pour les problèmes mnk-landscapes, on remarque que la valeur du degré de non-linéarité k peut avoir un impact sur les performances. En fait, ces différences de performances ne sont pas liées à la rugosité du problème, mais plus à l'ordre de Walsh utilisé. En effet, on peut observer que l'ordre de Walsh donnant les meilleurs rangs pour une instance est tel que $\mathcal{O} = k + 1$. Pour les instances MUBQP, parce qu'il s'agit de problèmes quadratiques, nous obtenons systématiquement de meilleures performances

avec un ordre de Walsh $\mathcal{O} = 2$. Cette observation n'est pas surprenante, puisque la performance des algorithmes est étroitement liée à la capacité du méta-modèle de Walsh à approximer correctement le paysage, capacité qui est directement liée au choix de l'ordre de Walsh. Nous pouvons donc conclure que le rang obtenu dans ces tableaux en utilisant un optimiseur du méta-modèle avec une stratégie de sélection donnée, change de manière homogène avec le choix de l'ordre. Ceci laisse penser qu'il existe un paramètre optimal qui dépend de la rugosité du paysage. Il peut être intéressant de constater que les rangs semblent varier également en fonction du budget disponible. Cela indique que les différentes variantes du framework S-MCO peuvent présenter des profils de convergence différents. Ce point sera analysé plus en détail dans les sections suivantes, lorsque nous étudierons le paramétrage (fixe ou dynamique) de l'ordre de Walsh.

La seconde observation que l'on peut faire à partir de ces tableaux se situe au niveau des optimiseurs. En effet, on peut voir très clairement que la stratégie de sélection qui fournit les meilleurs rangs est différente pour chaque optimiseur. Autrement dit, pour obtenir des performances optimales, la stratégie de sélection dans le framework S-MCO doit être obligatoirement choisie en fonction de l'optimiseur considéré. Plus précisément, en ce qui concerne les deux optimiseurs à base de recherche locale, c'est-à-dire $\mathsf{Optim}_{\mathsf{MLS}}$ et $\mathsf{Optim}_{\mathsf{PLS}}$, nous pouvons constater qu'ils sont plus performants lorsqu'ils utilisent la stratégie $\mathsf{Select}_{\mathsf{BI}_{\mathsf{norm}}}$ pour les instances de mnk-landscapes avec $k = 0$ et $k = 1$, alors que l'utilisation de la stratégie $\mathsf{Select}_{\mathsf{local}}$ est un choix légèrement meilleur pour ces deux optimiseurs pour les instances avec $k = 2$. Pour MUBQP, $\mathsf{Optim}_{\mathsf{PLS}}$ est plus performant avec la stratégie $\mathsf{Select}_{\mathsf{local}}$, alors que $\mathsf{Optim}_{\mathsf{MLS}}$ obtient ces meilleurs résultats avec la stratégie $\mathsf{Select}_{\mathsf{BI}_{\mathsf{norm}}}$. Concernant l'optimiseur $\mathsf{Optim}_{\mathsf{MOEA/D}}$, $\mathsf{Select}_{\mathsf{BI}}$ est la meilleure stratégie sur les instances de mnk-landscapes alors que $\mathsf{Select}_{\mathsf{local}}$ est à privilégier pour MUBQP.

Pour résumer ces dépendances, nous présentons dans le Tableau 4.6 une vue globale sur les performances des différentes stratégies de sélection en fonction des différents optimiseurs assistés par méta-modèle. Ce tableau est obtenu en triant les différentes stratégies selon les rangs moyens des Tableaux 4.2, 4.3, 4.4 et 4.5. En analysant l'impact de la stratégie de sélection sur les différents optimiseurs, il est intéressant de remarquer que la stratégie $\mathsf{Select}_{\mathsf{BI}_{\mathsf{norm}}}$ est préférée pour $\mathsf{Optim}_{\mathsf{PLS}}$ et $\mathsf{Optim}_{\mathsf{MLS}}$ avec un rang moyen de respectivement 1.40 et 1.29, alors que le rang moyen avec l'optimiseur $\mathsf{Optim}_{\mathsf{MOEA/D}}$ est de 4.36. Autrement dit, la stratégie $\mathsf{Select}_{\mathsf{BI}_{\mathsf{norm}}}$ peut être vue comme une bonne stratégie de sélection, sauf quand elle est utilisée en combinaison avec $\mathsf{Optim}_{\mathsf{MOEA/D}}$. En plus de tout cela, on remarque que la stratégie $\mathsf{Select}_{\mathsf{global}}$ ne semble convenir à

Tableau 4.6 – Classement des stratégies de sélection par rangs moyen ($L = \mathsf{Select}_{\mathsf{local}}$, $G = \mathsf{Select}_{\mathsf{global}}$, $B_C = \mathsf{Select}_{\mathsf{BI}}$, $B_N = \mathsf{Select}_{\mathsf{BI}_{\mathsf{norm}}}$).

		$\mathsf{Optim}_{\mathsf{MOEA/D}}$	$\mathsf{Optim}_{\mathsf{MLS}}$	$\mathsf{Optim}_{\mathsf{PLS}}$
mnk-landscapes	$k = 0$	$L \prec B_C \prec B_N \prec G$	$B_N \prec B_C \prec L \prec G$	$B_N \prec L \prec B_C \prec G$
mnk-landscapes	$k = 1$	$B_C \prec L \prec B_N \prec G$	$B_N \prec B_C \prec L \prec G$	$B_N \prec L \prec B_C \prec G$
mnk-landscapes	$k = 2$	$B_C \prec L \prec B_N \prec G$	$L \prec B_C \prec B_N \prec G$	$L \prec B_C \prec B_N \prec G$
MUBQP		$L \prec B_C \prec G \prec B_N$	$B_N \prec B_C \prec L \prec G$	$L \prec B_N \prec B_C \prec G$

aucun des optimiseurs étudiés, car elle nous donne les moins bons résultats pour chacun d'entre eux.

Pour la suite de notre étude sur les composants du framework S-MCO, nous utiliserons, si ce n'est pas précisé dans le texte, les stratégies de sélection permettant d'avoir les meilleurs résultats pour chacun des optimiseurs, comme indiqué dans le Tableau 4.6.

4.3.3 Analyse des optimiseurs du méta-modèle

Les Figures 4.2 et 4.3 nous montrent les profils de convergence des différentes variantes de S-MCO pour respectivement des instances de mnk-landscapes et MUBQP. Nous visons ici à analyser la qualité d'approximation du front Pareto obtenue avec les différents optimiseurs étudiés, et avec les différents réglages statiques de l'ordre de Walsh.

Comme première observation, nous pouvons voir que $\mathsf{Optim}_{\mathsf{MLS}}$ et $\mathsf{Optim}_{\mathsf{PLS}}$ ont des profils de convergence significativement meilleurs que $\mathsf{Optim}_{\mathsf{MOEA/D}}$. Ce constat est homogène pour toutes les instances étudiées, à quelques exceptions près. Il est intéressant de rappeler ici que le framework S-MCO utilise la décomposition à la fois pour structurer la population, et aussi pour sélectionner la solution suivante à évaluer, indépendamment de l'optimiseur considéré. D'où cette première observation, qui montre donc que le framework S-MCO n'a pas besoin d'être configuré avec un optimiseur interne basé sur la décomposition pour parcourir l'espace de recherche à l'aide du méta-modèle. Pour pousser plus loin notre discussion, nous pouvons constater que la seule raison qui peut motiver l'utilisation de MOEA/D comme optimiseur interne est sa complexité de calcul extrêmement efficace. En effet, comme on peut le voir sur la Figure 4.3, lorsqu'on étudie la plus grande instance MUBQP avec $n = 50$, et en utilisant l'ordre de Walsh le plus exigeant en termes de calcul $\mathcal{O} = 3$, le framework S-MCO est capable de terminer toutes les itérations (jusqu'à 1 500 évaluations réelles) dans le temps maximum autorisé par CPU, uniquement lorsqu'il est configuré avec l'optimiseur $\mathsf{Optim}_{\mathsf{MOEA/D}}$. Cette observation nous indique donc que $\mathsf{Optim}_{\mathsf{MOEA/D}}$ peut

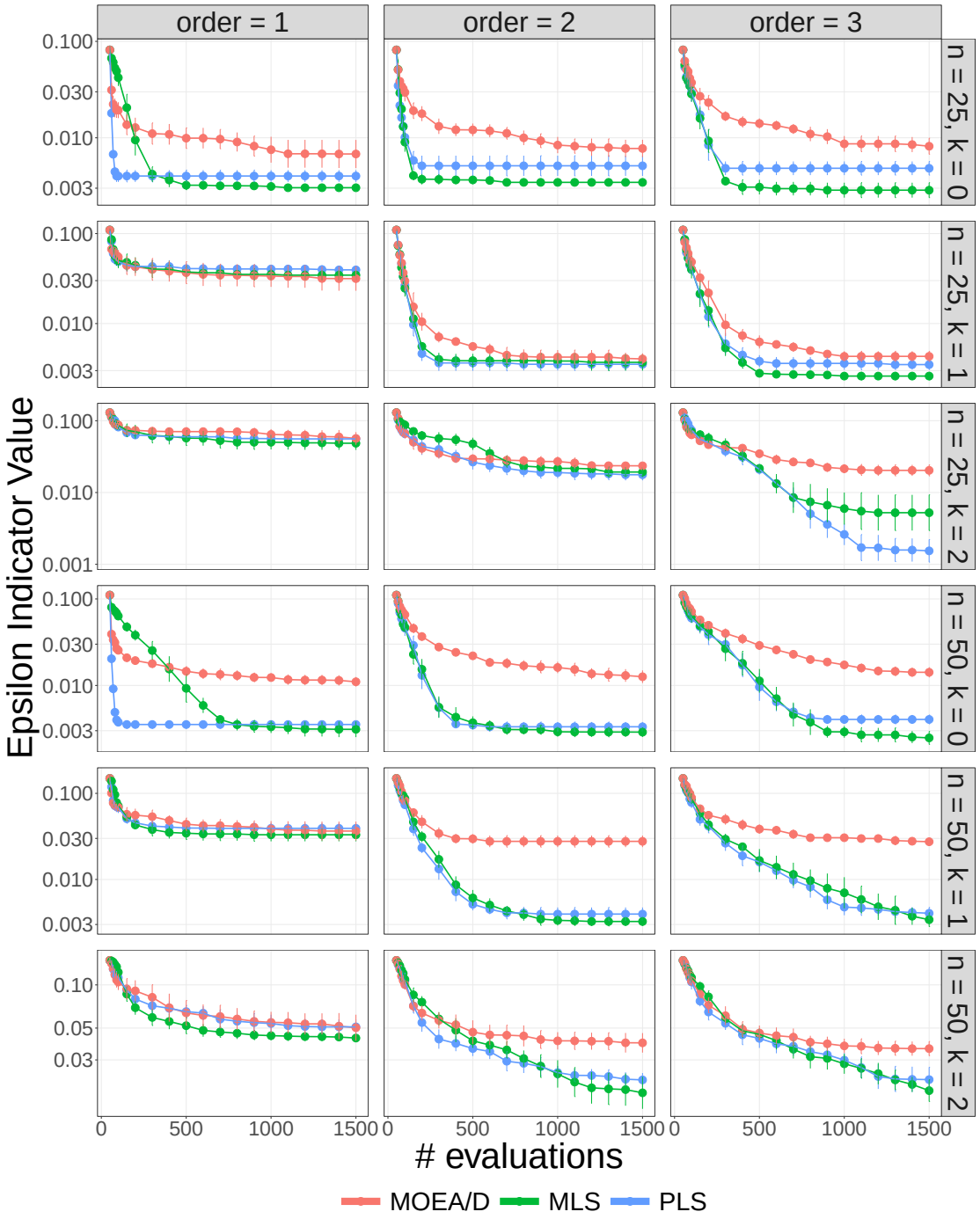

FIGURE 4.2 – Profils de convergence de S-MCO avec différents optimiseurs assistés par méta-modèles combinés à leur meilleure stratégie de sélection sur des instances de mnk-landscapes.

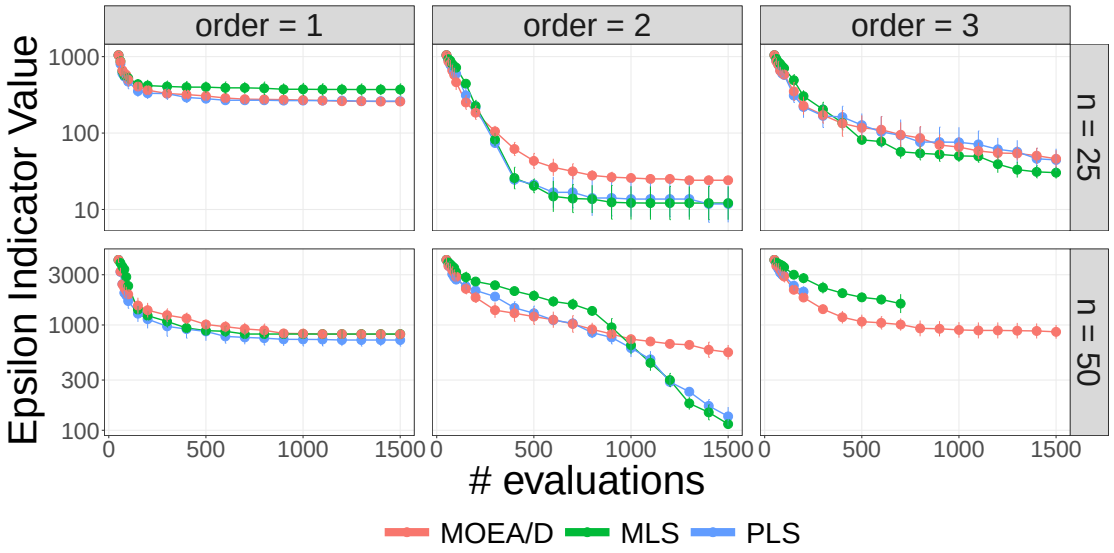

FIGURE 4.3 – Profils de convergence de S-MCO avec différents optimiseurs assistés par méta-modèles combinés à leur meilleure stratégie de sélection sur des instances de MUBQP.

être un choix raisonnable s'il existe un quelconque compromis sur le temps de calcul CPU par exécution entre le cout de la fonction d'évaluation et le cout de l'optimiseur du méta-modèle.

Analysons maintenant plus en détail $\text{Optim}_{\text{MLS}}$ et $\text{Optim}_{\text{PLS}}$, qui se sont avérés permettre une meilleure qualité d'approximation du front Pareto. Même si dans l'ensemble, ils affichent un comportement similaire, leur profil de convergence dépend à la fois de la difficulté du problème abordé, mais surtout de la complexité du méta-modèle de Walsh, c'est-à-dire le choix de l'ordre de Walsh. Les paragraphes suivants expliquent ce phénomène en divisant l'ensemble de nos instances étudiées en trois classes distinctes : linéaire, c'est-à-dire les instances de mnk-landscapes avec $k = 0$, quadratique, c'est-à-dire les instances de mnk-landscapes avec $k = 1$ et MUBQP, et finalement cubique, c'est-à-dire les instances de mnk-landscapes avec $k = 2$. On notera ici que l'ordre de Walsh optimal en théorie pour décomposer correctement ces fonctions est de respectivement 1, 2 et 3. Dans un contexte boite-noire, cette information n'est évidemment pas disponible. Dans le contexte de notre étude, ceci nous permet néanmoins d'analyser le comportement des optimiseurs avec des ordres optimaux (en théorie) ou non-optimaux.

Pour les instances linéaires, $\text{Optim}_{\text{PLS}}$ utilisé en combinaison avec l'ordre de Walsh $\mathcal{O} = 1$ (ordre optimal pour les instances linéaires), a un meilleur profil de convergence que $\text{Optim}_{\text{MLS}}$, on peut voir en effet que l'optimiseur $\text{Optim}_{\text{PLS}}$ converge très rapidement

vers une approximation de haute qualité, alors que $\text{Optim}_{\text{MLS}}$ n'est capable d'obtenir une meilleure qualité que lorsqu'un plus grand nombre d'évaluations est disponible. Lorsqu'on utilise un ordre de Walsh surestimé de 2 ou 3, la situation est clairement inversée, et $\text{Optim}_{\text{MLS}}$ s'avère nettement plus performant que $\text{Optim}_{\text{PLS}}$, indépendamment du budget disponible.

Pour les instances quadratiques, en fixant l'ordre de Walsh à l'ordre optimal $\mathcal{O} = 2$, $\text{Optim}_{\text{MLS}}$ et $\text{Optim}_{\text{PLS}}$ présentent le même profil de convergence pour les instances avec $n = 25$. En revanche, quand $n = 50$, $\text{Optim}_{\text{PLS}}$ est légèrement meilleur avec un budget restreint, tandis que $\text{Optim}_{\text{MLS}}$ est légèrement meilleur avec des budgets plus élevés, ce qui rappelle leur comportement pour les instances linéaires. Lorsque l'ordre de Walsh est fixé à une valeur sous-estimée $\mathcal{O} = 1$, la qualité de l'approximation obtenue avec tous les optimiseurs chute par rapport à un ordre (optimal) de 2. Il est intéressant de noter que l'utilisation d'une valeur d'ordre sous-estimée n'empêche pas le framework S-MCO de continuer à s'améliorer au cours des toutes premières itérations, c'est-à-dire avec un budget très restreint. Lorsque l'ordre de Walsh est fixé à une valeur surestimée $\mathcal{O} = 3$, la qualité d'approximation globale des deux optimiseurs semble être comparable à celle de l'ordre exact, ce qui semble cohérent, car tous les coefficients présents avec un ordre $\mathcal{O} = 2$ sont également présents avec un ordre $\mathcal{O} = 3$.

Enfin, pour les instances cubiques les plus complexes, nous avons constaté que pour les instances avec $n = 25$, un ordre sous-estimé de 1 ou 2 n'est pas compétitif par rapport à l'ordre exact de 3, indépendamment de l'optimiseur utilisé. Pour les problèmes de dimensions $n = 50$, le framework S-MCO est très clairement peu performant pour un ordre sous-estimé $\mathcal{O} = 1$. Cependant, la situation s'améliore en utilisant un ordre $\mathcal{O} = 2$ pour $\text{Optim}_{\text{PLS}}$ et $\text{Optim}_{\text{MLS}}$, ce qui indique que pour des problèmes aussi difficiles, un ordre de Walsh suffisamment grand, mais non exact, permet quand même d'obtenir une approximation raisonnable.

À partir de nos observations, nous pouvons tirer deux conclusions générales. Premièrement, les deux optimiseurs de recherche locale $\text{Optim}_{\text{MLS}}$ et $\text{Optim}_{\text{PLS}}$ sont efficaces dans l'exploration de l'espace de recherche à l'aide du méta-modèle de Walsh. Deuxièmement, nous confirmons qu'un optimiseur du méta-modèle seul n'est pas capable de fournir de bonnes performances indépendamment du budget disponible et du paramétrage de l'ordre de Walsh. Nous allons maintenant étudier l'impact de l'ordre de Walsh, qui constitue le dernier composant du framework S-MCO.

4.3.4 Étude du choix de l'ordre de Walsh

Dans cette section, nous étudions l'impact du paramétrage de l'ordre de Walsh à travers une analyse de trois stratégies différentes. Nous étudierons ce paramétrage uniquement pour $\mathsf{Optim}_{\mathsf{MLS}}$ et $\mathsf{Optim}_{\mathsf{PLS}}$ étant donné qu'ils surpassent les résultats de $\mathsf{Optim}_{\mathsf{MOEA/D}}$. Les Figures 4.4 et 4.5 nous montrent les profils de convergence du framework S-MCO avec différents paramétrages de l'ordre de Walsh pour la stratégie $\mathsf{Select}_{\mathsf{BI}_{\mathsf{norm}}}$ sur des instances de MUBQP et mnk-landscapes, respectivement. Avec ces figures, on comprend très clairement que sous-estimer la valeur de l'ordre (relativement à l'ordre optimal) a un impact extrêmement négatif sur les performances générales du framework S-MCO. En outre, surestimer la valeur de l'ordre ne permet pas toujours d'atteindre des résultats compétitifs avec les meilleures variantes de S-MCO pour un budget et une instance donnée. Cela peut paraître surprenant à première vue, car on pourrait penser que surestimer l'ordre ne peut que rendre le méta-modèle de Walsh plus complexe sans en diminuer la précision. Même si cette affirmation semble vraie en théorie, l'augmentation de l'ordre de Walsh rend le modèle beaucoup plus difficile à ajuster et à optimiser avec précision en pratique. Cette observation est illustrée par la Figure 4.6, qui montre l'erreur absolue moyenne du méta-modèle de Walsh en fonction du nombre de solutions dans la base d'apprentissage en considérant l'optimiseur $\mathsf{Optim}_{\mathsf{MLS}}$ combiné à la stratégie $\mathsf{Select}_{\mathsf{BI}_{\mathsf{norm}}}$. Nous voyons clairement que la surestimation de l'ordre peut conduire à une moins bonne qualité d'apprentissage, en fonction du degré de non-linéarité du problème et de la taille de la base d'apprentissage. En effet, plus le modèle est complexe, et plus le nombre de solutions dans la base d'apprentissage doit être important pour que la régression Lasso, que nous utilisons dans nos expériences, réussisse à trouver un bon ajustement des coefficients du méta-modèle de Walsh.

En comparaison à un réglage statique de l'ordre de Walsh, nous pouvons observer que les deux stratégies dynamiques fournissent clairement un meilleur choix par rapport à la stratégie statique. En effet, elles s'avèrent très compétitives, bien qu'il n'y ait pas de séparation nette entre la stratégie $\mathsf{Order}_{\mathsf{greedy}}$ et la stratégie $\mathsf{Order}_{\mathsf{random}}$. En effet, la stratégie $\mathsf{Order}_{\mathsf{greedy}}$ n'est pas meilleure que la stratégie $\mathsf{Order}_{\mathsf{random}}$, indépendamment de l'optimiseur utilisé, de l'instance étudiée et du budget disponible. Par exemple, nous pouvons voir que la stratégie $\mathsf{Order}_{\mathsf{greedy}}$ surpasse la stratégie $\mathsf{Order}_{\mathsf{random}}$ pour les instances MUBQP de taille $n = 50$. Cependant, cela n'est pas plus vrai pour $n = 25$ et l'optimiseur $\mathsf{Optim}_{\mathsf{PLS}}$. De même, la stratégie $\mathsf{Order}_{\mathsf{greedy}}$ obtient un meilleur profil de convergence que la stratégie $\mathsf{Order}_{\mathsf{random}}$ en utilisant $\mathsf{Optim}_{\mathsf{PLS}}$ pour les instances de mnk-landscapes avec $n = 50$ et $k = 0$, alors que cela n'est plus vrai en combinant la

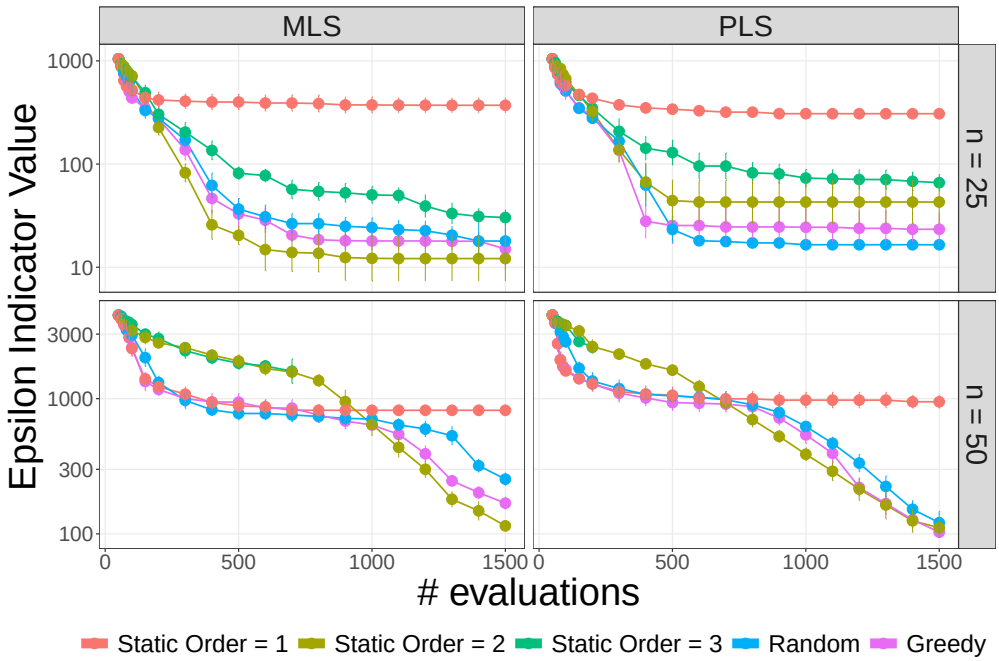

FIGURE 4.4 – Profils de convergence du framework S-MCO avec différents paramétrages de l'ordre de Walsh pour la stratégie $\text{Select}_{\text{BI}_{\text{norm}}}$ sur des instances de MUBQP.

stratégie $\text{Order}_{\text{greedy}}$ avec $\text{Optim}_{\text{MLS}}$.

Pour résumer, nous constatons que les deux stratégies dynamiques de paramétrage de l'ordre de Walsh obtiennent des résultats très compétitifs. Une stratégie d'ordre dynamique est toujours meilleure qu'un ordre statique sous-estimé et peut être dans la plupart des cas meilleure qu'un ordre statique surestimé. De façon surprenante, une stratégie dynamique peut même être meilleure qu'une stratégie statique utilisant l'ordre de Walsh optimal en théorie pour certaines instances, par exemple pour mnk-landscapes avec $n = 50$ et $k = 1$. Ces résultats suggèrent que des stratégies dynamiques plus sophistiquées pour définir l'ordre de Walsh méritent d'être étudiées à l'avenir.

4.3.5 Comparaison entre le framework S-MCO et des algorithmes non-assistés par méta-modèle

Cette section permet de conclure notre analyse en étudiant les performances du framework S-MCO avec des algorithmes qui ne sont pas assistées par méta-modèle. Les algorithmes non-assistés par méta-modèle que nous considérons sont spécifiquement les optimiseurs $\text{Optim}_{\text{MLS}}$, $\text{Optim}_{\text{PLS}}$ et $\text{Optim}_{\text{MOEA/D}}$ qui sont expérimentés dans cette section en tant que tels, c'est-à-dire indépendamment du framework S-MCO et en

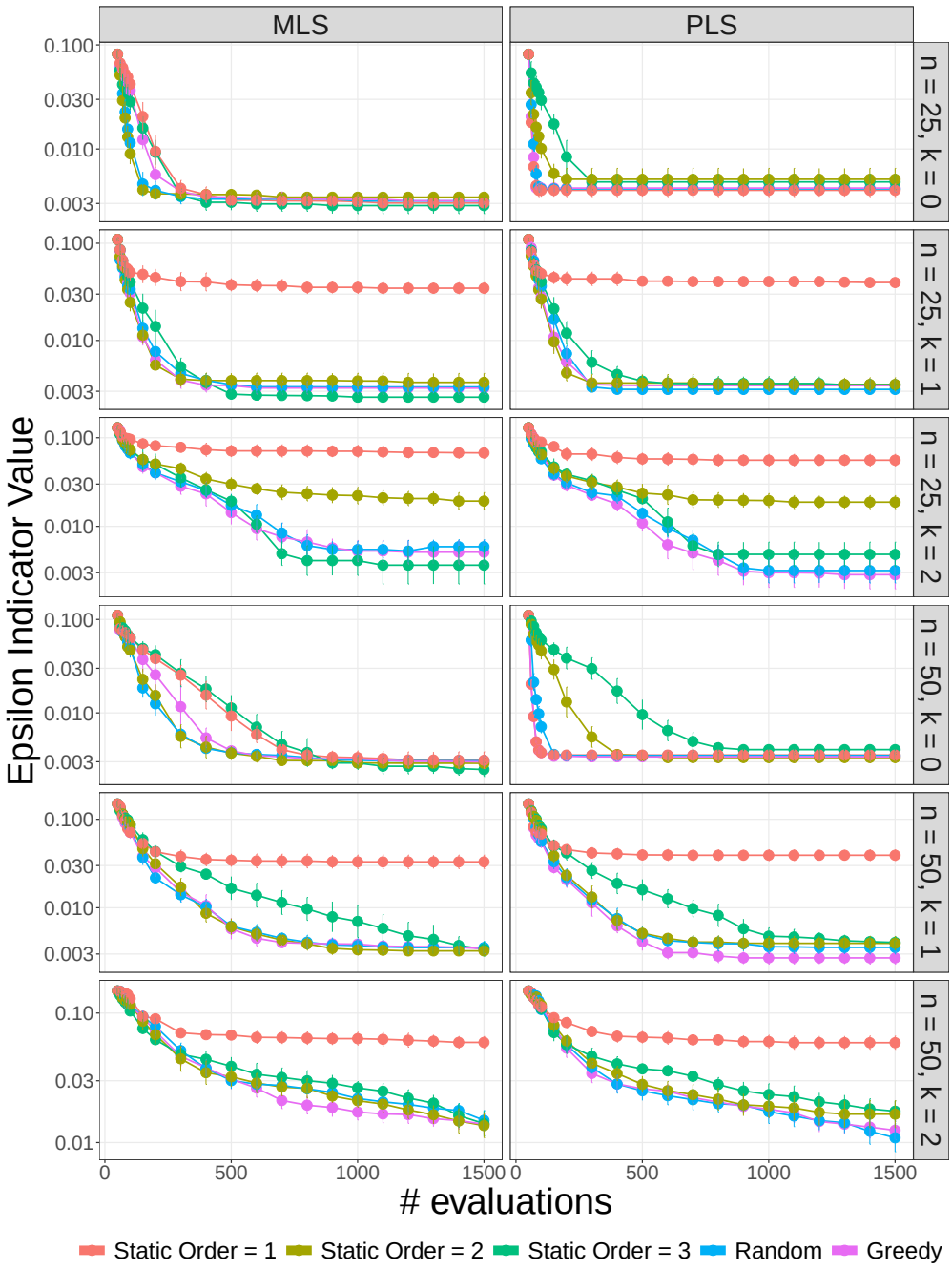

FIGURE 4.5 – Profils de convergence du framework S-MCO avec différents paramétrages de l'ordre de Walsh pour la stratégie $\text{Select}_{\text{BI}_{\text{norm}}}$ sur des instances de mnk-landscapes.

utilisant directement les fonctions coûteuses pour évaluer les solutions.

Les résultats sont donnés dans la Figure 4.7 où le framework S-MCO est configuré

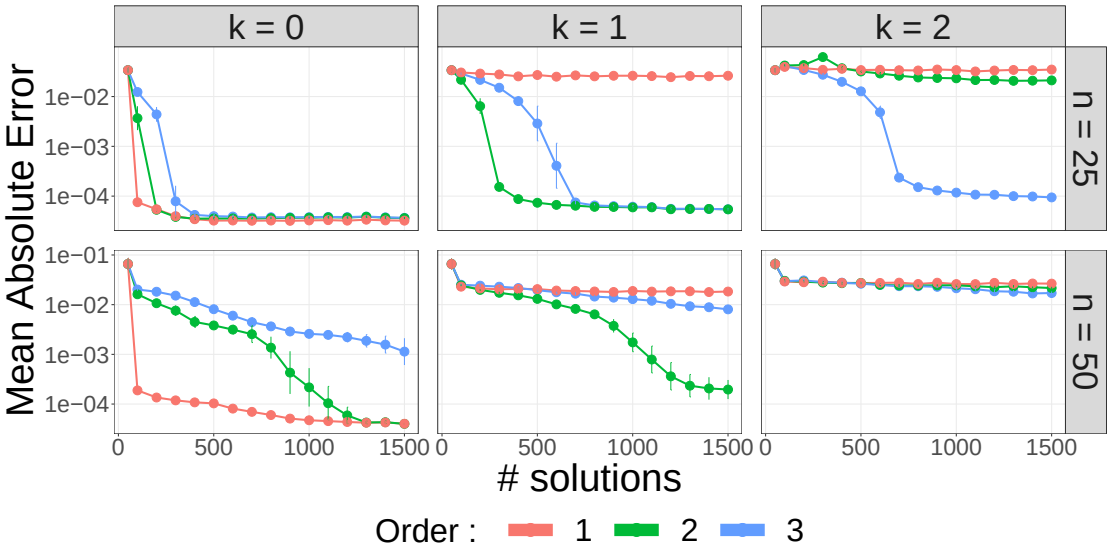

FIGURE 4.6 – Mean absolute error (MAE) obtenu par le méta-modèle de Walsh en fonction du nombre de solutions dans la base d'apprentissage avec l'optimiseur $\text{Optim}_{\text{MLS}}$ combiné à la stratégie $\text{Select}_{\text{Bl}_{\text{norm}}}$ pour les ordres $\mathcal{O} \in \{1, 2, 3\}$ sur mnk-landscapes.

avec la stratégie $\text{Select}_{\text{Bl}_{\text{norm}}}$ pour choisir les solutions à évaluer, ainsi qu'avec la stratégie $\text{Order}_{\text{greedy}}$ pour choisir dynamiquement l'ordre de Walsh. Cette configuration est celle qui donne de bons résultats sur l'ensemble des instances étudiées.

Dans la Figure 4.7, on peut voir très clairement que toutes les variantes de S-MCO, obtiennent des résultats significativement meilleurs, indépendamment de l'optimiseur utilisé, de l'instance étudiée ou du budget disponible. On peut également voir que le choix de l'optimiseur à utiliser dans S-MCO ne doit pas se faire en fonction de ses performances en tant qu'algorithme non-assisté par méta-modèle. En effet, on peut voir que malgré les très mauvais résultats d'$\text{Optim}_{\text{MLS}}$ sans l'utilisation du méta-modèle, cet algorithme donne parmi les meilleurs résultats quand il est combiné aux bons composants dans S-MCO.

4.4 Conclusion

Dans ce chapitre, nous avons proposé une étude autour de la conception d'un framework modulaire assisté par méta-modèle pour l'optimisation combinatoire multi-objectif coûteuse. À notre connaissance, il s'agit de la première étude approfondie dans ce domaine. Grâce à une analyse empirique détaillée, nous avons fourni une étude méthodique des effets combinés des différents composants du framework. En outre,

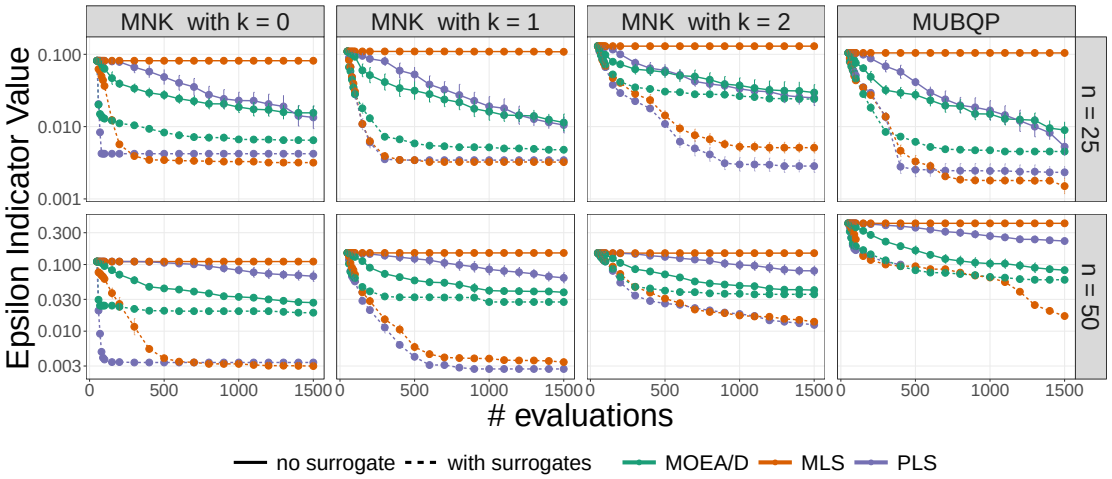

FIGURE 4.7 – Comparaison des approches sans méta-modèle avec S-MCO en utilisant la stratégie $\text{Order}_{\text{greedy}}$ et en combinaison de la stratégie $\text{Select}_{\text{BI}_{\text{norm}}}$.

nous avons constaté qu'il existe une interaction non triviale entre la stratégie permettant d'optimiser les méta-modèles, celle utilisée pour remplacer les objectifs coûteux et la stratégie utilisée pour la sélection de la prochaine solution à évaluer.

Dans un contexte où l'on décompose un problème multi-objectif en plusieurs sous-problèmes mono-objectifs, nous montrons qu'une stratégie de sélection basée sur les améliorations prédites par un méta-modèle des solutions candidates est très efficace. Nous montrons dans ce même contexte que les optimiseurs à base de recherche locale sont plus performants pour traiter l'optimisation interne du méta-modèle discret de Walsh. Enfin, nous avons souligné l'importance du paramétrage de l'ordre de Walsh et nous avons étudié des stratégies dynamiques simples qui se sont révélées capables de s'adapter à un ensemble de problèmes d'optimisation boite-noire avec différents degrés de non-linéarité.

Bien que le framework S-MCO proposé se soit révélé extrêmement efficace par rapport aux algorithmes évolutionnaires multi-objectif sans assistance par méta-modèle, un certain nombre de questions restent ouvertes. Par exemple, il serait intéressant d'utiliser le framework S-MCO pour gérer des stratégies de sélection basées sur d'autres paradigmes de recherche multi-objectif. Notre framework est basé sur la décomposition pour structurer la population de solutions évaluées, et pour estimer quelle nouvelle solution enfant est la plus prometteuse pour une véritable évaluation (coûteuse). Un autre défi consiste à aborder d'autres domaines d'optimisation combinatoire, tels que les problèmes de permutation, pour lesquels d'autres méta-modèles mono-objectif existent

dans la littérature spécialisée [73, 112]. La question principale est donc d'étudier dans quelle mesure le framework S-MCO peut fonctionner efficacement dans de tels contextes. Notez que, en principe, le framework S-MCO devrait fonctionner avec n'importe quel méta-modèle mono-objectif. Cependant, l'étude des effets combinés de ses composants ne peut être fonction que d'un domaine d'optimisation cible. Nous espérons que l'étude approfondie menée dans ce chapitre accélérera les fondations d'une méthodologie unifiée pour la conception et l'analyse d'algorithmes d'optimisation multi-objectif assistés par méta-modèle.

Conclusion générale

Cette thèse porte sur l'étude de l'optimisation combinatoire multi-objectif boite-noire à travers les méthodes d'optimisation basées sur des techniques de décomposition. Le but premier de cette thèse est d'enrichir le framework état-de-l'art MOEA/D avec de nouveaux composants algorithmiques, et d'en étudier l'impact sur le processus de recherche évolutionnaire. Nos travaux se concentrent spécifiquement sur les problèmes combinatoires multi-objectifs, ce qui est en soi un domaine relativement moins bien étudié, surtout dans le contexte des algorithmes à base de décomposition.

Synthèse des contributions

Optimisation combinatoire multi-objectif. Dans le Chapitre 1, nous avons introduit les principes de l'optimisation combinatoire multi-objectif. Cette introduction nous a permis de rappeler les concepts de base permettant une bonne compréhension de ce manuscrit, tout en introduisant les deux problèmes combinatoires multi-objectif qui ont été considérés dans nos analyses expérimentales (mnk-landscapes et MUBQP). Ce chapitre nous a permis de présenter chacun des composants de l'algorithme état-de-l'art MOEA/D que nous considérons comme la base de nos différentes contributions. Ce chapitre nous a également donné l'occasion de présenter le framework logiciel que nous avons développé pendant ces trois dernières années pour aider la communauté à implémenter et à expérimenter facilement de nouvelles variantes de MOEA/D, ou tout autre algorithme basé sur la décomposition.

Sélection des sous-problèmes mono-objectifs. Le Chapitre 2 a été consacré à l'étude de l'un des composants majeurs dans les algorithmes évolutionnaires par décomposition, qui consiste à déterminer les sous-problèmes (mono-objectifs) à optimiser à chacune des générations du processus évolutionnaire. Nous avons proposé une analyse basée sur l'étude de trois paramètres clés que sont la taille de la population (ou le nombre de

sous-problèmes définis dans l'algorithme), le nombre de sous-problèmes optimisés par génération, et finalement la méthode permettant de choisir ces sous-problèmes. Notre étude expérimentale a montré qu'il ne faut pas simplement s'intéresser à la manière de sélectionner les sous-problèmes comme l'ont fait les nombreuses études sur ce domaine, mais que le nombre de sous-problèmes sélectionnés est tout aussi important pour améliorer les performances de l'algorithme, notamment son comportement "anytime".

Échappement aux optimums locaux. Le Chapitre 3 a permis l'étude de mécanismes permettant de s'échapper des optima locaux dans MOEA/D. Deux sous-composants ont été étudiés : un composant de détection des optima locaux, puis un composant de perturbation de la population. L'étude expérimentale effectuée dans ce chapitre a prouvé que l'utilisation de tels composants dans MOEA/D est extrêmement bénéfique pour la qualité de l'approximation. Ce composant a également montré qu'il pouvait compléter les performances de nos travaux du Chapitre 2, notamment pour attaquer efficacement les instances les plus difficiles.

Optimisation combinatoire multi-objectif coûteuse. Finalement, le Chapitre 4 de ce mémoire a été consacré à l'optimisation coûteuse. Relativement peu d'études existent sur ce sujet en optimisation combinatoire, et encore moins en optimisation combinatoire multi-objectif. Ainsi, nous avons proposé un framework de décomposition inspiré de MOEA/D pour optimiser des problèmes coûteux en se basant sur le méta-modèle de Walsh. Grâce à une étude expérimentale approfondie, qui est à notre connaissance la première étude complète dans ce domaine, nous avons fourni une analyse méthodique des différents composants proposés en fonction des caractéristiques du problème à résoudre.

Perspectives

Les travaux réalisés dans le cadre de cette thèse nous ont permis de contribuer aux fondations d'un framework basé sur la décomposition pour des problèmes combinatoires multi-objectifs. Ces travaux suggèrent de nouvelles perspectives dont quelques-unes sont présentées ci-dessous.

Sélection des sous-problèmes dans S-MCO. Nos différents travaux ont tous été présentés sous une forme modulaire, afin de faciliter l'utilisation et la configuration des différents composants que nous avons pu proposer. Il serait intéressant d'utiliser le

composant de sélection des sous-problèmes étudié dans le Chapitre 2 pour améliorer le comportement anytime du framework S-MCO dans un contexte d'optimisation coûteuse. En effet, le fait de permettre à l'algorithme de converger plus rapidement grâce à ce composant peut être très bénéfique dans un contexte où le nombre d'évaluations total est limité.

Configuration automatique des paramètres. Au cours de nos travaux, nous avons proposé de nouveaux composants permettant une configuration plus fine du comportement de nos algorithmes. Les paramètres de ces nouveaux composants ont été étudiés en considérant des plages de valeur assez larges et représentatives, ceci afin de fournir une aide aux utilisateurs souhaitant configurer ces paramètres sur le même type de problèmes. Il serait intéressant d'étudier des méthodes de paramétrage automatiques, pour à la fois réduire l'effort de configuration, mais aussi améliorer les performances de l'algorithme. À titre d'exemple, nous pensons qu'une approche adaptative, qui agit en continu au cours du processus de recherche, permettrait d'améliorer la configuration de MOEA/D-(μ, λ, sps) (Chapitre 2), ainsi que le choix de taille de la fenêtre permettant de détecter les optima locaux (Chapitre 3), qui s'avère être dépendante du budget disponible. De même, dans le Chapitre 4, nous avons proposé deux stratégies simples montrant l'intérêt de modifier l'ordre de Walsh au cours du processus de recherche. Il serait intéressant de tester d'autres stratégies adaptatives basées sur de nouveaux critères comme la qualité du modèle par exemple.

Étude de nouveaux problèmes. Nous nous sommes focalisés dans ce manuscrit sur l'étude d'un framework de décomposition pour des problèmes combinatoires. Nous considérons nos travaux comme une autre étape dans la mise en place d'un framework visant à optimiser des problèmes multi-objectifs de tout type avec des techniques de décomposition. Il serait intéressant de réviser nos travaux pour des problèmes continus afin de comparer les comportements que nous avons pu constater dans le cadre de problèmes combinatoires. Nous sommes convaincus que, même s'il peut y avoir des différences de comportement sur certains composants, une telle étude permettrait d'améliorer nos connaissances globales des méthodes basées sur le concept de décomposition en optimisation multi-objectif.

Par ailleurs, tous les défis sont loin d'être résolus en ce qui concerne l'optimisation combinatoire. On pourrait par exemple étudier l'impact de la corrélation des objectifs sur nos différents algorithmes, ce qui à notre avis pourra se traduire par de nouveaux composants ou de nouvelles variantes de nos composants. De la même manière, il serait

intéressant d'étudier les mécanismes d'échappement aux optima locaux pour des problèmes avec un nombre plus grand d'objectifs. Enfin, il serait intéressant d'expérimenter l'algorithme S-MCO (Chapitre 4) sur des problèmes plus complexes, que cela soit en augmentant le nombre d'objectifs ou en augmentant la dimension des solutions.

Bibliographie

[1] Hernán E. Aguirre et Kiyoshi Tanaka. « Working principles, behavior, and performance of MOEAs on MNK-landscapes ». In : *European Journal of Operational Research* 181.3 (sept. 2007), p. 1670-1690.

[2] Khalil Amine. « Multiobjective Simulated Annealing : Principles and Algorithm Variants ». In : *Advances in Operations Research* 2019 (mai 2019), p. 1-13.

[3] Anne Auger. « Benchmarking the (1+1) evolution strategy with one-fifth success rule on the BBOB-2009 function testbed ». In : *Proceedings of the 11th annual conference companion on Genetic and evolutionary computation conference - GECCO '09*. Montreal, Québec, Canada : ACM Press, 2009, p. 2447.

[4] S. Bandyopadhyay, S. Saha, U. Maulik et K. Deb. « A Simulated Annealing-Based Multiobjective Optimization Algorithm : AMOSA ». In : *IEEE Transactions on Evolutionary Computation* 12.3 (juin 2008), p. 269-283.

[5] Ricardo Baptista et Matthias Poloczek. « Bayesian Optimization of Combinatorial Structures ». In : *arXiv :1806.08838 [cs, math, stat]* (oct. 2018). arXiv : 1806.08838.

[6] Thomas Bartz-Beielstein et Martin Zaefferer. « Model-based methods for continuous and discrete global optimization ». In : *Applied Soft Computing* 55 (juin 2017), p. 154-167.

[7] Nicolas Berveglieri, Bilel Derbel, Arnaud Liefooghe, Hernán Aguirre, Qingfu Zhang et Kiyoshi Tanaka. « Designing parallelism in surrogate-assisted multiobjective optimization based on decomposition ». In : *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. New York, NY, USA : Association for Computing Machinery, juin 2020, p. 462-470.

[8] N. Beume, B. Naujoks et M. Emmerich. « SMS-EMOA : Multiobjective selection based on dominated hypervolume ». In : *Eur. J. Oper. Res.* 181.3 (2007), p. 1653-1669.

[9] Francesco Biscani et Dario Izzo. « A parallel global multiobjective framework for optimization : pagmo ». In : *Journal of Open Source Software* 5.53 (sept. 2020), p. 2338.

[10] Julian Blank et Kalyanmoy Deb. « Pymoo : Multi-Objective Optimization in Python ». In : *IEEE Access* 8 (2020), p. 89497-89509.

[11] Pedro Manuel F. C. Borges et Michael Pilegaard Hansen. « A basis for future success in multiobjective combinatorial optimization problems ». In : (1998).

[12] Christina Brester, Ivan Ryzhikov et Eugene Semenkin. « On Performance Improvement Based on Restart Meta-Heuristic Implementation for Solving Multiobjective Optimization Problems ». In : *Advances in Swarm Intelligence*. Sous la dir. d'Ying Tan, Hideyuki Takagi, Yuhui Shi et Ben Niu. T. 10386. Cham : Springer International Publishing, 2017, p. 23-30.

[13] Xinye Cai, Zhiwei Mei, Zhun Fan et Qingfu Zhang. « A Constrained Decomposition Approach With Grids for Evolutionary Multiobjective Optimization ». In : *IEEE Transactions on Evolutionary Computation* 22.4 (août 2018), p. 564-577.

[14] Felipe Campelo, Lucas S. Batista et Claus Aranha. « The MOEADr Package : A Component-Based Framework for Multiobjective Evolutionary Algorithms Based on Decomposition ». In : *Journal of Statistical Software* 92.6 (2020).

[15] Z. Cao, K. Liu et B. Hu. « An Improved MOEA/D for Order Scheduling Problem in Automated Warehouse ». In : *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. ISSN : 2161-8089. Août 2018, p. 797-802.

[16] V. Černý. « Thermodynamical approach to the traveling salesman problem : An efficient simulation algorithm ». In : *Journal of Optimization Theory and Applications* 45.1 (jan. 1985), p. 41-51.

[17] T. Chiang et Y. Lai. « Improving MOEA/D by an adaptive mating selection mechanism ». In : *2011 IEEE Congress of Evolutionary Computation (CEC)*. 2011.

[18] A. Colorni, M. Dorigo, V. Maniezzo, F. Varela et P. Bourgine. « Distributed Optimization by Ant Colonies ». In : *undefined* (1992).

[19] John A. Cornell. « Some Comments on Designs for Cox's Mixture Polynomial ». In : *Technometrics* 17.1 (1975), p. 25-35.

[20] Piotr Czyżżak et Adrezej Jaszkiewicz. « Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization ». In : *Journal of Multi-Criteria Decision Analysis* 7.1 (1998), p. 34-47.

[21] K. Deb et H. Jain. « An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I : Solving Problems With Box Constraints ». In : *IEEE Transactions on Evolutionary Computation* 18.4 (août 2014), p. 577-601.

[22] K. Deb, A. Pratap, S. Agarwal et T. Meyarivan. « A fast and elitist multiobjective genetic algorithm : NSGA-II ». In : *IEEE Transactions on Evolutionary Computation* 6.2 (avr. 2002). tex.ids= debFastElitistMultiobjective2002a conferenceName : IEEE Transactions on Evolutionary Computation, p. 182-197.

[23] Bilel Derbel, Geoffrey Pruvost et Byung-Woo Hong. « Enhancing Moea/d with Escape Mechanisms ». In : *2021 IEEE Congress on Evolutionary Computation (CEC)*. Kraków, Poland : IEEE, juin 2021, p. 1163-1170.

[24] Elizabeth D. Dolan et Jorge J. Moré. « Benchmarking optimization software with performance profiles ». In : *Mathematical Programming* 91.2 (jan. 2002), p. 201-213.

[25] Nelson Dunford et Jacob T. Schwartz. *Linear operators*. Wiley Classics Library ed. New York : Interscience Publishers, 1988.

[26] Bradley Efron, Trevor Hastie, Iain Johnstone et Robert Tibshirani. « Least angle regression ». In : *The Annals of Statistics* 32.2 (avr. 2004).

[27] Matthias Ehrgott. *Multicriteria optimization*. 2nd ed. Berlin : Springer, 2005.

[28] Xavier Gandibleux, Nazik Mezdaoui et Arnaud Fréville. « A Tabu Search Procedure to Solve MultiObjective Combinatorial Optimization Problems ». In : *Advances in Multiple Objective and Goal Programming*. Sous la dir. de G. Fandel, W. Trockel, Rafael Caballero, Francisco Ruiz et Ralph Steuer. T. 455. Berlin, Heidelberg : Springer Berlin Heidelberg, 1997, p. 291-300.

[29] Fred Glover et Manuel Laguna. « Tabu Search ». In : *Handbook of Combinatorial Optimization*. Sous la dir. de Ding-Zhu Du et Panos M. Pardalos. Boston, MA : Springer US, 1998, p. 2093-2229.

[30] P. L. Hammer et S. Rudeanu. *Boolean Methods in Operations Research and Related Areas*. Berlin Heidelberg : Springer-Verlag, 1968.

[31] Peter L. Hammer et Eliezer Shlifer. « Applications of pseudo-Boolean methods to economic problems ». In : *Theory and Decision* 1.3 (mar. 1971), p. 296-308.

[32] R. W. Hamming. « Error Detecting and Error Correcting Codes ». In : *Bell System Technical Journal* 29.2 (avr. 1950), p. 147-160.

[33] K. Harada, S. Hiwa et T. Hiroyasu. « Adaptive weight vector assignment method for MOEA/D ». In : *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. Nov. 2017, p. 1-9.

[34] Trevor Hastie, Robert Tibshirani et Martin Wainwright. *Statistical Learning with Sparsity : The Lasso and Generalizations*. 1re éd. Chapman et Hall/CRC, 2015.

[35] Siegfried Helbig et Darinka Pateva. « On several concepts for epsilon-efficiency ». In : *Operations-Research-Spektrum* 16.3 (sept. 1994), p. 179-186.

[36] Robert J Howlett et Lakhmi C Jain. *Radial Basis Function Networks 2 : New Advances in Design*. OCLC : 1159962738. 0.

[37] Robert J. Howlett et L. C. Jain, éd. *Radial basis function networks 1 : recent developments in theory and applications*. vol. 66. Heidelberg ; New York : Physica-Verlag, 2001.

[38] Hisao Ishibuchi, Ryo Imada, Naoki Masuyama et Yusuke Nojima. « Two-Layered Weight Vector Specification in Decomposition-Based Multi-Objective Algorithms for Many-Objective Optimization Problems ». In : *2019 IEEE Congress on Evolutionary Computation (CEC)*. Wellington, New Zealand : IEEE, juin 2019, p. 2434-2441.

[39] Hisao Ishibuchi, Yuji Sakane, Noritaka Tsukamoto et Yusuke Nojima. « Adaptation of Scalarizing Functions in MOEA/D : An Adaptive Scalarizing Function-Based Multiobjective Evolutionary Algorithm ». In : *Evolutionary Multi-Criterion Optimization*. Sous la dir. de Matthias Ehrgott, Carlos M. Fonseca, Xavier Gandibleux, Jin-Kao Hao et Marc Sevaux. T. 5467. Berlin, Heidelberg : Springer Berlin Heidelberg, 2009, p. 438-452.

[40] Hisao Ishibuchi, Yuji Sakane, Noritaka Tsukamoto et Yusuke Nojima. « Simultaneous use of different scalarizing functions in MOEA/D ». In : *Proceedings of the 12th annual conference on Genetic and evolutionary computation - GECCO '10*. Portland, Oregon, USA : ACM Press, 2010, p. 519.

[41] Hisao Ishibuchi, Yu Setoguchi, Hiroyuki Masuda et Yusuke Nojima. « How to compare many-objective algorithms under different settings of population and archive sizes ». In : *2016 IEEE Congress on Evolutionary Computation (CEC)*. Vancouver, BC, Canada : IEEE, juil. 2016, p. 1149-1156.

[42] E. Jiang et L. Wang. « A Modified MOEA/D for Energy-efficient Flexible Job Shop Scheduling Problem ». In : *2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. ISSN : 2157-362X. Déc. 2018, p. 1476-1480.

[43] S. Jiang et L. Zhang. « Energy-Oriented Scheduling for Hybrid Flow Shop With Limited Buffers Through Efficient Multi-Objective Optimization ». In : *IEEE Access* 7 (2019), p. 34477-34487.

[44] Shouyong Jiang, Shengxiang Yang, Yong Wang et Xiaobin Liu. « Scalarizing Functions in Decomposition-Based Multiobjective Evolutionary Algorithms ». In : *IEEE Transactions on Evolutionary Computation* 22.2 (avr. 2018), p. 296-313.

[45] D. Johnson et L. A. McGeoch. « The Traveling Salesman Problem : A Case Study in Local Optimization ». In : *undefined* (2008).

[46] S. A. Kauffman. *The origins of order*. Oxford University Press, 1993.

[47] Ke Li, Qingfu Zhang, Sam Kwong, Miqing Li et Ran Wang. « Stable Matching-Based Selection in Evolutionary Multiobjective Optimization ». In : *IEEE Transactions on Evolutionary Computation* 18.6 (2014), p. 909-923.

[48] Hans Kellerer, Ulrich Pferschy et David Pisinger. *Knapsack Problems*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2004.

[49] J. Kennedy et R. Eberhart. « Particle swarm optimization ». In : *Proceedings of ICNN'95 - International Conference on Neural Networks*. T. 4. Perth, WA, Australia : IEEE, 1995, p. 1942-1948.

[50] Yong-Hyuk Kim, Alberto Moraglio, Ahmed Kattan et Yourim Yoon. « Geometric Generalisation of Surrogate Model-Based Optimisation to Combinatorial and Program Spaces ». In : *Mathematical Problems in Engineering* 2014 (2014), p. 1-10.

[51] S. Kirkpatrick, C. D. Gelatt et M. P. Vecchi. « Optimization by Simulated Annealing ». In : *Science* 220.4598 (mai 1983), p. 671-680.

[52] Gary Kochenberger, Jin-Kao Hao, Fred Glover, Mark Lewis, Zhipeng Lü, Haibo Wang et Yang Wang. « The unconstrained binary quadratic programming problem : a survey ». In : *Journal of Combinatorial Optimization* 28.1 (juil. 2014), p. 58-81.

[53] D. J. Laughhunn. « Quadratic Binary Programming with Application to Capital-Budgeting Problems ». In : *Operations Research* 18.3 (juin 1970), p. 454-461.

[54] Yuri Lavinas, Claus Aranha et Marcelo Ladeira. « Improving Resource Allocation in MOEA/D with Decision-Space Diversity Metrics ». In : *Theory and Practice of Natural Computing*. Sous la dir. de Carlos Martín-Vide, Geoffrey Pond et Miguel A. Vega-Rodríguez. T. 11934. Series Title : Lecture Notes in Computer Science. Cham : Springer International Publishing, 2019, p. 134-146.

[55] Yuri Lavinas, Claus Aranha et Testuya Sakurai. « Using diversity as a priority function for resource allocation on MOEA/D ». In : *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. Prague Czech Republic : ACM, juil. 2019, p. 215-216.

[56] Florian Leprêtre, Cyril Fonlupt, Sébastien Verel, Virginie Marion, Rolando Armas, Hernán Aguirre et Kiyoshi Tanaka. « Fitness landscapes analysis and adaptive algorithms design for traffic lights optimization on SIALAC benchmark ». In : *Applied Soft Computing* 85 (déc. 2019), p. 105869.

[57] Florian Leprêtre, Sébastien Verel, Cyril Fonlupt et Virginie Marion. « Walsh functions as surrogate model for pseudo-boolean optimization problems ». In : *Proceedings of the Genetic and Evolutionary Computation Conference on - GECCO '19*. Prague, Czech Republic : ACM Press, 2019, p. 303-311.

[58] H. Li et D. Landa-Silva. « An Adaptive Evolutionary Multi-Objective Approach Based on Simulated Annealing ». In : *Evolutionary Computation* 19.4 (déc. 2011), p. 561-595.

[59] H. Li et Q. Zhang. « MOEA/D-DE : Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II ». In : *IEEE Transactions on Evolutionary Computation* 13.2 (avr. 2009), p. 284-302.

[60] Z. Li, L. He et Y. Chu. « An Improved Decomposition Multiobjective Optimization Algorithm with Weight Vector Adaptation Strategy ». In : *2017 13th International Conference on Semantics, Knowledge and Grids (SKG)*. Août 2017, p. 19-24.

[61] Arnaud Liefooghe et Bilel Derbel. « A Correlation Analysis of Set Quality Indicator Values in Multiobjective Optimization ». In : *Genetic and Evolutionary Computation Conference (GECCO 2016)*. Denver, United States, juil. 2016.

[62] Arnaud Liefooghe, Sébastien Verel, Luís Paquete et Jin-Kao Hao. « Experiments on Local Search for Bi-objective Unconstrained Binary Quadratic Programming ». In : *Evolutionary Multi-Criterion Optimization*. Sous la dir. d'António Gaspar-Cunha, Carlos Henggeler Antunes et Carlos Coello Coello. Cham : Springer International Publishing, 2015, p. 171-186.

[63] J. Liu, M. Gong, Q. Miao, X. Wang et H. Li. « Structure Learning for Deep Neural Networks Based on Multiobjective Optimization ». In : *IEEE Transactions on Neural Networks and Learning Systems* 29.6 (juin 2018), p. 2450-2463.

[64] Yuan Liu, Yikun Hu, Ningbo Zhu, Kenli Li, Juan Zou et Miqing Li. « A Decomposition-based Multiobjective Evolutionary Algorithm with Weights Updated Adaptively ». In : *Information Sciences* (avr. 2021).

[65] Manuel López-Ibáñez, Luís Paquete et Thomas Stützle. « Exploratory Analysis of Stochastic Local Search Algorithms in Biobjective Optimization ». In : *Experimental Methods for the Analysis of Optimization Algorithms*. Sous la dir. de Thomas Bartz-Beielstein, Marco Chiarandini, Luís Paquete et Mike Preuss. Berlin, Heidelberg : Springer Berlin Heidelberg, 2010, p. 209-222.

[66] Helena R. Lourenço, Olivier C. Martin et Thomas Stützle. « Iterated Local Search ». In : *Handbook of Metaheuristics*. Sous la dir. de Fred Glover et Gary A. Kochenberger. T. 57. Boston : Kluwer Academic Publishers, 2003, p. 320-353.

[67] Xiaoliang Ma, Qingfu Zhang, Guangdong Tian, Junshan Yang et Zexuan Zhu. « On Tchebycheff Decomposition Approaches for Multiobjective Evolutionary Optimization ». In : *IEEE Transactions on Evolutionary Computation* 22.2 (avr. 2018), p. 226-244.

[68] Jesús Mejía. *Metaheuristics.jl : High performance metaheuristics for optimization purely coded in Julia*. Juin 2021.

[69] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller et Edward Teller. « Equation of State Calculations by Fast Computing Machines ». In : *The Journal of Chemical Physics* 21.6 (juin 1953), p. 1087-1092.

[70] Kaisa Miettinen. *Nonlinear Multiobjective Optimization*. OCLC : 863690021. 1998.

[71] Alberto Moraglio. « Towards a geometric unification of evolutionary algorithms ». Ph.D. University of Essex, 2008.

[72] Alberto Moraglio et Ahmed Kattan. « Geometric Generalisation of Surrogate Model Based Optimisation to Combinatorial Spaces ». In : *Evolutionary Computation in Combinatorial Optimization*. Sous la dir. de Peter Merz et Jin-Kao Hao. T. 6622. Berlin, Heidelberg : Springer Berlin Heidelberg, 2011, p. 142-154.

[73] Alberto Moraglio, Yong-Hyuk Kim et Yourim Yoon. « Geometric surrogate-based optimisation for permutation-based problems ». In : *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation - GECCO '11*. Dublin, Ireland : ACM Press, 2011, p. 133.

[74] T. Murata et H. Ishibuchi. « MOGA : multi-objective genetic algorithms ». In : *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*. T. 1. Perth, WA, Australia : IEEE, 1995, p. 289.

[75] Antonio J. Nebro, Juan J. Durillo et Matthieu Vergne. « Redesigning the jMetal Multi-Objective Optimization Framework ». In : *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. Madrid Spain : ACM, juil. 2015, p. 1093-1100.

[76] Yew-Soon Ong, Meng Lim et Xianshun Chen. « Memetic Computation—Past, Present & Future [Research Frontier ». In : *IEEE Computational Intelligence Magazine* 5.2 (mai 2010), p. 24-31.

[77] Christos H. Papadimitriou et Kenneth Steiglitz. *Combinatorial Optimization : Algorithms and Complexity*. Courier Corporation, jan. 1998.

[78] Luis Paquete, Marco Chiarandini et Thomas Stützle. « Pareto Local Optimum Sets in the Biobjective Traveling Salesman Problem : An Experimental Study ». In : *Metaheuristics for Multiobjective Optimisation*. Sous la dir. de G. Fandel, W. Trockel, Xavier Gandibleux, Marc Sevaux, Kenneth Sörensen et Vincent T'kindt. T. 535. Berlin, Heidelberg : Springer Berlin Heidelberg, 2004, p. 177-199.

[79] Luis Paquete, Tommaso Schiavinotto et Thomas Stützle. « On local optima in multiobjective combinatorial optimization problems ». In : *Annals of Operations Research* 156.1 (sept. 2007), p. 83-97.

[80] Luis Paquete et Thomas Stützle. « A study of stochastic local search algorithms for the biobjective QAP with correlated flow matrices ». In : *European Journal of Operational Research* 169.3 (mar. 2006), p. 943-959.

[81] Geoffrey Pruvost, Bilel Derbel et Arnaud Liefooghe. « Moead-framework : a modular MOEA/D python framework ». In : *Under review : Journal of Open Source Software* ().

[82] Geoffrey Pruvost, Bilel Derbel, Arnaud Liefooghe, Ke Li et Qingfu Zhang. « On the Combined Impact of Population Size and Sub-problem Selection in MOEA/D ». In : *Evolutionary Computation in Combinatorial Optimization*. T. 12102. Series Title : Lecture Notes in Computer Science. Cham : Springer International Publishing, 2020, p. 131-147.

[83] Geoffrey Pruvost, Bilel Derbel, Arnaud Liefooghe, Sébastien Verel et Qingfu Zhang. « Surrogate-assisted multi-objective combinatorial optimization based on decomposition and walsh basis ». In : *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. Cancún Mexico : ACM, juin 2020, p. 542-550.

[84] Geoffrey Pruvost, Bilel Derbel, Arnaud Liefooghe, Sébastien Verel et Qingfu Zhang. *A Modular Surrogate-assisted Framework for Expensive Multiobjective Combinatorial Optimization*. Oct. 2021.

[85] Anthony Przybylski et Xavier Gandibleux. « Multi-objective branch and bound ». In : *European Journal of Operational Research* 260.3 (août 2017), p. 856-872.

[86] Yutao Qi, Xiaoliang Ma, Fang Liu, Licheng Jiao, Jianyong Sun et Jianshe Wu. « MOEA/D with Adaptive Weight Adjustment ». In : *Evolutionary Computation* 22.2 (juin 2014), p. 231-264.

[87] Qingfu Zhang et Hui Li. « MOEA/D : A Multiobjective Evolutionary Algorithm Based on Decomposition ». In : *IEEE Transactions on Evolutionary Computation* 11.6 (déc. 2007), p. 712-731.

[88] Qingfu Zhang, Wudong Liu, Edward Tsang et Botond Virginas. « Expensive Multiobjective Optimization by MOEA/D With Gaussian Process Model ». In : *IEEE Transactions on Evolutionary Computation* 14.3 (juin 2010), p. 456-474.

[89] Juan P. Romero, Angel Ibeas, Jose L. Moura, Juan Benavente et Borja Alonso. « A Simulation-optimization Approach to Design Efficient Systems of Bike-sharing ». In : *Procedia - Social and Behavioral Sciences* 54 (oct. 2012), p. 646-655.

[90] Henry Scheffe. « The Simplex-Centroid Design for Experiments with Mixtures ». In : *Journal of the Royal Statistical Society. Series B (Methodological)* 25.2 (1963), p. 235-263.

[91] Henry Scheffé. « Experiments with Mixtures ». In : *journal of the Royal Statistical Society : Series B* 20.2 (juil. 1958), p. 344-360.

[92] J. Siwei, C. Zhihua, Z. Jie et O. Yew-Soon. « Multiobjective optimization by decomposition with Pareto-adaptive weight vectors ». In : *2011 Seventh International Conference on Natural Computation*. T. 3. ISSN : 2157-9563. Juil. 2011, p. 1260-1264.

[93] K.I. Smith, R.M. Everson, J.E. Fieldsend, C. Murphy et R. Misra. « Dominance-Based Multiobjective Simulated Annealing ». In : *IEEE Transactions on Evolutionary Computation* 12.3 (juin 2008), p. 323-342.

[94] Y. Sun, G. G. Yen et Z. Yi. « IGD Indicator-Based Evolutionary Algorithm for Many-Objective Optimization Problems ». In : *IEEE Transactions on Evolutionary Computation* 23.2 (avr. 2019), p. 173-187.

[95] Y. Tan, Y. Jiao et X. Wang. « MOEA/D with Uniform Design for the Multiobjective 0/1 Knapsack Problem ». In : *2011 Seventh International Conference on Computational Intelligence and Security*. Déc. 2011, p. 96-100.

[96] Ryoji Tanabe et Hisao Ishibuchi. « An analysis of control parameters of MOEA/D under two different optimization scenarios ». In : *Applied Soft Computing* 70 (sept. 2018), p. 22-40.

[97] Robert Tibshirani. « Regression Shrinkage and Selection via the Lasso ». In : *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996), p. 267-288.

[98] Sébastien Verel, Bilel Derbel, Arnaud Liefooghe, Hernán Aguirre et Kiyoshi Tanaka. « A Surrogate Model Based on Walsh Decomposition for Pseudo-Boolean Functions ». In : *Parallel Problem Solving from Nature – PPSN XV*. Sous la dir. d'Anne Auger, Carlos M. Fonseca, Nuno Lourenço, Penousal Machado, Luís Paquete et Darrell Whitley. T. 11102. Cham : Springer International Publishing, 2018, p. 181-193.

[99] Tobias Wagner, Heike Trautmann et Luis Martí. « A Taxonomy of Online Stopping Criteria for Multi-Objective Evolutionary Algorithms ». In : *Evolutionary Multi-Criterion Optimization*. Sous la dir. de Ricardo H. C. Takahashi, Kalyanmoy Deb, Elizabeth F. Wanner et Salvatore Greco. T. 6576. Berlin, Heidelberg : Springer Berlin Heidelberg, 2011, p. 16-30.

[100] Jia Wang, Yuchao Su, Qiuzhen Lin, Lijia Ma, Dunwei Gong, Jianqiang Li et Zhong Ming. « A survey of decomposition approaches in multiobjective evolutionary algorithms ». In : *Neurocomputing* 408 (sept. 2020). tex.ids= wangSurveyDecompositionApproaches2020a, p. 308-330.

[101] Luping Wang, Qingfu Zhang, Aimin Zhou, Maoguo Gong et Licheng Jiao. « Constrained Subproblems in a Decomposition-Based Multiobjective Evolutionary Algorithm ». In : *IEEE Transactions on Evolutionary Computation* 20.3 (juin 2016), p. 475-480.

[102] Peng Wang, Wen Zhu, Haihua Liu, Bo Liao, Lijun Cai, Xiaohui Wei, Siqi Ren et Jialiang Yang. « A new resource allocation strategy based on the relationship between subproblems for MOEA/D ». In : *Information Sciences* 501 (2019), p. 337-362.

[103] Frank Wilcoxon. « Individual Comparisons by Ranking Methods ». In : *Biometrics Bulletin* 1.6 (déc. 1945), p. 80.

[104] Carsten Witt. « Population size versus runtime of a simple evolutionary algorithm ». In : *Theoretical Computer Science* 403.1 (août 2008), p. 104-120.

[105] C Witzgall. *Mathematical methods of site selection for Electronic Message Systems (EMS)*. NASA STI/Recon Technical Report N. 1975.

[106] Jianyou Xu, Chin-Chia Wu, Yunqiang Yin et Win-Chin Lin. « An iterated local search for the multi-objective permutation flowshop scheduling problem with sequence-dependent setup times ». In : *Applied Soft Computing* 52 (mar. 2017), p. 39-47.

[107] Q. Xu, Z. Xu et T. Ma. « A Survey of Multiobjective Evolutionary Algorithms Based on Decomposition : Variants, Challenges and Future Directions ». In : *IEEE Access* 8 (2020), p. 41588-41614.

[108] Li-Yuan Xue, Rong-Qiang Zeng, Yang Wang et Ming-Sheng Shang. « Solving Bi-objective Unconstrained Binary Quadratic Programming Problem with Multi-objective Backbone Guided Search Algorithm ». In : *Intelligent Computing Theories and Application*. Sous la dir. de De-Shuang Huang et Kang-Hyun Jo. Cham : Springer International Publishing, 2016, p. 745-753.

[109] Yee Leung, Yong Gao et Zong-Ben Xu. « Degree of population diversity - a perspective on premature convergence in genetic algorithms and its Markov chain analysis ». In : *IEEE Transactions on Neural Networks* 8.5 (sept. 1997), p. 1165-1176.

[110] Yourim Yoon, Yong-Hyuk Kim, Alberto Moraglio et Byung-Ro Moon. « Quotient geometric crossovers and redundant encodings ». In : *Theoretical Computer Science* 425 (mar. 2012), p. 4-16.

[111] Martin Zaefferer. « Surrogate models for discrete optimization problems ». Thèse de doct. 2018.

[112] Martin Zaefferer, Jörg Stork et Thomas Bartz-Beielstein. « Distance Measures for Permutations in Combinatorial Efficient Global Optimization ». In : *Parallel Problem Solving from Nature – PPSN XIII*. Sous la dir. de Thomas Bartz-Beielstein, Jürgen Branke, Bogdan Filipič et Jim Smith. T. 8672. Cham : Springer International Publishing, 2014, p. 373-383.

[113] Martin Zaefferer, Jörg Stork, Martina Friese, Andreas Fischbach, Boris Naujoks et Thomas Bartz-Beielstein. « Efficient global optimization for combinatorial problems ». In : *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. Vancouver BC Canada : ACM, juil. 2014, p. 871-878.

[114] Murilo Zangari, Roberto Santana, Alexander Mendiburu et Aurora Pozo. « On the Design of Hard mUBQP Instances ». In : *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. New York, NY, USA : Association for Computing Machinery, juil. 2016, p. 421-428.

[115] Saul Zapotecas-Martínez, Hernán E. Aguirre, Kiyoshi Tanaka et Carlos A. Coello Coello. « On the low-discrepancy sequences and their use in MOEA/D for high-dimensional objective spaces ». In : *2015 IEEE Congress on Evolutionary Computation (CEC)*. ISSN : 1089-778X, 1941-0026. 2015, p. 2835-2842.

[116] H. Zhang, D. Yue, W. Yue, K. Li et M. Yin. « MOEA/D-Based Probabilistic PBI Approach for Risk-Based Optimal Operation of Hybrid Energy System With Intermittent Power Uncertainty ». In : *IEEE Transactions on Systems, Man, and Cybernetics : Systems* 51.4 (avr. 2021), p. 2080-2090.

[117] Qingfu Zhang, Wudong Liu et Hui Li. « The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances ». In : *2009 IEEE Congress on Evolutionary Computation*. Trondheim, Norway : IEEE, mai 2009, p. 203-208.

[118] Wei Zheng, Yanyan Tan, Lili Meng et Huaxiang Zhang. « An improved MOEA/D design for many-objective optimization problems ». In : *Applied Intelligence* 48.10 (oct. 2018), p. 3839-3861.

[119] A. Zhou et Q. Zhang. « Are all the subproblems equally important ? Resource allocation in decomposition-based multiobjective evolutionary algorithms ». In : *IEEE Transactions on Evolutionary Computation* 20.1 (2016), p. 52-64.

[120] E. Zitzler et L. Thiele. « Multiobjective evolutionary algorithms : a comparative case study and the strength Pareto approach ». In : *IEEE Transactions on Evolutionary Computation* 3.4 (nov. 1999), p. 257-271.

[121] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca et V.G. da Fonseca. « Performance assessment of multiobjective optimizers : an analysis and review ». In : *IEEE Transactions on Evolutionary Computation* 7.2 (avr. 2003), p. 117-132.

[122] Eckart Zitzler, Joshua Knowles et Lothar Thiele. « Quality Assessment of Pareto Set Approximations ». In : *Multiobjective Optimization*. Sous la dir. de Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen et Roman Słowiński. T. 5252. Berlin, Heidelberg : Springer Berlin Heidelberg, 2008, p. 373-404.

Contributions à l'optimisation multi-objectif à base de décomposition

Résumé

Dans cette thèse, nous nous intéressons à l'optimisation combinatoire multi-objectif, et en particulier aux algorithmes évolutionnaires à base de décomposition. Ce type d'approches consiste à décomposer le problème multi-objectif original en plusieurs sous-problèmes mono-objectifs, qui sont alors résolus de façon coopérative. Dans ce cadre, nous considérons la conception et l'analyse de nouveaux composants algorithmiques contribuant à la mise en place des fondations d'un framework d'optimisation à base de décomposition pour les problèmes combinatoires multi-objectifs dits "boîte-noire", pour lesquels la forme analytique des fonctions objectif n'est pas connue de l'algorithme de résolution. Tout d'abord, nous étudions les éléments clés pour une meilleure répartition des efforts de calculs tout au long du processus d'optimisation. Pour cela, nous étudions l'impact conjoint de la taille de la population et du nombre de solutions générées par itération, tout en proposant différentes stratégies de sélection du ou des sous-problèmes à optimiser à chaque étape. Nous étudions ensuite différents mécanismes permettant de s'échapper des optima locaux. Ceux-ci s'inspirent de techniques issues de l'optimisation mono-objectif et permettent d'améliorer considérablement le profil de convergence des approches considérées. Nous nous plaçons pour finir dans un contexte d'optimisation coûteuse, où l'évaluation de chaque solution s'avère particulièrement gourmande en temps de calcul, ce qui limite considérablement le budget alloué à l'optimisation. Pour cela, nous étudions de nouveaux composants s'appuyant sur des méta-modèles combinatoires, et nous considérons leur intégration au sein d'approches évolutionnaires multi-objectifs basées sur la décomposition.

Mots clés : optimisation multi-objectif, optimisation combinatoire, décomposition, moea/d, méta-modèle, optimum local, algorithme évolutionnaire

---

Abstract

In this thesis, we are interested in multi-objective combinatorial optimization, and in particular in evolutionary algorithms based on decomposition. This type of approaches consists in decomposing the original multi-objective problem into multiple single-objective sub-problems that are then solved cooperatively. In this context, we consider the design and the analysis of new algorithmic components contributing to the establishment of the foundations of an optimization framework based on decomposition for multi-objective combinatorial problems known as "black box", i.e., for which the analytical form of the objective functions is not available to the solving algorithm. First of all, we investigate the key components for a better distribution of the computational efforts during the optimization process. To this end, we study the joint impact of the population size and of the number of solutions generated at each iteration, while proposing different strategies for selecting one ore multiple sub-problem(s) to be optimized at each stage. We then study different mechanisms allowing to escape from local optima. They are inspired by techniques from single-objective optimization, and we show they can significantly improve the convergence profile of the considered approaches. Finally, we consider the context of expensive optimization, where the evaluation of each solution is particularly intensive in terms of computational resources. This hence drastically restrict the budget allocated to the optimization process. As such, we investigate new components based on combinatorial meta-models, and we consider their integration within decomposition-based multi-objective evolutionary approaches.

Keywords: multi-objective optimisation, combinatorial optimization, decomposition, moea/d, surrogate, local optima, evolutionary algorithm

---