Ordonnancement de projet avec contraintes de ressources et aide à la décision multi-objectif

Xixi Wang

Thèse
de doctorat
de l'UTT

Xixi WANG

Ordonnancement de projet avec contraintes de ressources et aide à la décision multi-objectif

Spécialité :
Optimisation et Sûreté des Systèmes

2017TROYO022

Année 2017

utt
UNIVERSITÉ DE TECHNOLOGIE
TROYES

THESE

pour l'obtention du grade de

DOCTEUR de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

Spécialité : OPTIMISATION ET SURETE DES SYSTEMES

présentée et soutenue par

Xixi WANG

le 26 septembre 2017

Ordonnancement de projet avec contraintes de ressources et aide à la décision multi-objectif

JURY

M. L. AMODEO	PROFESSEUR DES UNIVERSITES	Président
M. M. BOUDHAR	PROFESSEUR	Examinateur
M. F. DUGARDIN	MAITRE DE CONFERENCES	Directeur de thèse
M. A. NAKIB	MAITRE DE CONFERENCES - HDR	Rapporteur
Mme N. SAUER	PROFESSEUR DES UNIVERSITES	Rapporteur
M. F. YALAOUI	PROFESSEUR DES UNIVERSITES	Directeur de thèse

Remerciements

Tout d'abord, j'exprime mes remerciements les plus sincères et mon profond respect à mes directeurs de thèse : Pr. Farouk Yalaoui et Dr. Frédéric Dugardin, d'avoir participé à ma formation universitaire et à mon apprentissage de la recherche, pour leur précieux encadrement, d'avoir su me guider avec attention et de m'avoir accordée leur confiance pour que je puisse mener bien à mes travaux de recherches. Je remercie Pr. Yalaoui pour son chaleureux accueil au sein du laboratoire d'optimisation des systèmes industriels.

J'adresse toute ma gratitude aux membres du jury qui font l'honneur de participer à l'examen de cette thèse : M. Amir Nakib, Maître de Conférences à l'Université Paris-Est-Créteil et Mme Nathalie Sauer, Professeur des Universités à l'Université Lorraine en leur qualité de rapporteurs, M. Lionel Amodeo, Professeur des Universités à l'Université de Technologie de Troyes et M. Mourad Boudhar, Professeur à l'Université des Sciences et de la Technologie Houari-Boumédiène en Algérie en leur qualité d'examinateurs.

Je tiens à remercier mes chers amis et collègues, Quy, Yipeng, Nan et Mohamed, doctorants à l'UTT pour leurs soutiens et leurs partages, qui m'ont apporté des réflexions et des inspirations.

Je remercie également Pr. Lionel Amodeo, Dr. Alice Yalaoui, Dr. Yassine Ouazene et Dr. Taha Arbaoui ainsi que les autres membres au LOSI pour leurs encouragements, leurs présences et leurs dévouements exemplaires.

J'aimerais adresser mes plus profonds remerciements à mes parents, qui m'ont toujours encouragée et soutenue tout au long de mon cursus en France. Ils ont su m'offrir les meilleures chances pour réussir et pour poursuivre ma carrière. Enfin, c'est avec grand plaisir que je salue Jiayu, Imane, Qingqing, Hajar, Chun, Jinrui... tous les amis qui m'ont offert leur soutien moral pendant cette expérience précieuse de ma vie.

Table des matières

Table des figures

Liste des tableaux

Introduction

De nos jours, la notion de gestion ou d'optimisation des resrouces devient incontournable dans la plupart des secteurs d'activités. La maîtrise des ressources est un sujet très présent dans les industries, car elle représente une des plus importantes problématiques au sein d'une entreprise.

Une entreprise se trouve en permanence face à des décisions pour assurer son bon fonctionnement. Pour diriger une société, il faut agir sur le long terme (niveau stratégique), le moyen terme (niveau tactique) et bien-sûr le court terme (niveau opérationnel). Au niveau stratégique, une société doit gérer son organisation, ses produits et services, de même que la gestion globale de ses ressources. Le niveau tactique s'appuie sur les décisions à moyen terme comme la prévision des demandes et la planification de la production. Finalement, les décisions opérationnelles sont prises de façon continue. Elles mettent l'accent sur les décisions détaillées que l'on envisage chaque jour, soit sur une ligne de production, soit dans un projet actif.

En effet, ces trois niveaux de décision se différentient non seulement sur la durée et les domaines de compétence, mais aussi sur leur enjeux dans une entreprise. Le niveau stratégique concerne les décisions dirigeant toute l'entreprise, le niveau tactique permet réaliser l'objectif issue des décisions stratégiques, alors que le niveau opérationnel demande des réactions plus flexibles et agiles. Si le niveau stratégique est la construction d'un bâtiment, le niveau tactique correspond aux piliers et les décisions opérationnelles font les briques.

Nos travaux portent sur le niveau opérationnel et ont pour objectif de fournir une aide à la décision. Plus précisément, nous considérons un problème d'ordonnancement de projet avec des contraintes de ressources (*Resource Constraint Project Scheduling Problem*, *RCPSP*). La résolution du problème consiste à ordonnancer les tâches du projet, en respectant les capacités limitées des ressources utilisées.

Le RCPSP est un des problèmes les plus étudiés dans la littérature de l'ordonnancement. Il soulève des problématiques intéressantes dans le monde de la recherche, et il concerne également les applications industrielles. Vis-à-vis des projets, les entreprises ont besoin d'agir rapidement afin de garantir un bon état d'avancement et respecter les contraintes. Les ressources peuvent être

consommables ou renouvelables. Elles sont soumises à des limitation de la capacité. Par exemple, le budget attribué à un projet peut être considéré comme une ressource *consommable*, car l'argent disponible pour le projet est dépensé au cours du temps. Au contraire, les personnes impliquées dans un projet sont plutôt des ressources *renouvelables* car elles sont disponibles pendant chaque période de temps.

Lorsque l'on considère un problème de type RCPSP, la durée du projet (aussi appelée "makespan") est mise en avant dans la plupart des études car elle montre directement la productivité. Cependant, chercher simplement à réduire le makespan ne satisfait pas forcément les entreprises, puisqu'elles peuvent faire face à d'autres problèmes comme gérer les retards importants, les blocages d'avancement à cause des ressources insuffisantes, etc. C'est pourquoi nous considérons dans cette thèse un problème multi-objectif, et nous cherchons des solutions tenant compte de plusieurs aspects.

Plus précisément, trois objectifs sont considérés dans nos travaux. Dans un premier temps, nous cherchons à minimiser la durée du projet afin d'assurer une bonne performance en terme de productivité. Ensuite, nous introduisons le retard total des tâches comme une deuxième critère, dont la minimisation permet de finir les tâches dans le délai souhaité. Ce critère reflète la qualité des services et il est lié à l'indice de satisfaction des clients. Pour finir, on s'intéresse à l'allocation des ressources et on cherche les solutions avec meilleur équilibrage de ressources.

L'objectif de cette thèse est donc de développer des méthodes performantes pour résoudre le problème décrit ci-dessus et de proposer aux décideurs les solutions qui optimisent conjointement les trois critères.

Le premier chapitre de ce mémoire est destiné à présenter une revue de la littérature, du problème générique du RCPSP ainsi que des problèmes multi-objectif. Certaines méthodes de résolution des problèmes multi-objectifs, y compris les méthodes exactes et les méthodes approchées, sont également synthétisées. Enfin, un résumé sur les métriques d'évaluation des solutions est présenté.

Dans le Chapitre 2, nous considérons un problème bi-objectif où le makespan et le retard total des tâches sont optimisés. Les formulations mathématiques et deux méthodes exactes sont d'abord présentées. Cependant, la résolution exacte ne permet de résoudre que les problèmes de petites tailles en temps limité en raison de son aspect NP-difficile. Pour cette raison, un algorithme génétique, le *Non-dominated Sorting Genetic Algorithm II* [19] (NSGAII) est implémenté afin d'améliorer la vitesse d'exécution. Ensuite, deux métriques d'évaluation sont calculées pour conclure la qualité des solutions proposées proposées par la métaheuristique.

Un troisième objectif, l'équilibrage d'allocation de ressources, est introduit dans le Chapitre 3. Naturellement, avec ces trois objectifs, la nature du problème d'optimisation change et nous avons adapté les méthodes de résolution en conséquence. Le NSGAII est adapté dans un premier temps pour résoudre le problème tri-objectif. Nous utilisons aussi le NSGAIII [18], une version plus récente que NSGAII, pour trouver les solutions approchées. Par la suite, une recherche locale utilisant le technique de *Mapping* [4] est également mise en œuvre pour mieux explorer de l'espace de recherche. Plusieurs stratégies d'exploration sont considérées. Les expérimentations dans ce chapitre portent d'abord sur une comparaison entre le NSGAII et NSGAIII. La recherche locale est ensuite intégrée dans chacune des deux métaheuristiques pour améliorer la qualité des solutions. Nous proposons également une amélioration plus adaptée à notre problème et nous comparons les résultats avec le schéma d'origine de la méthode de Mapping [4].

Le Chapitre 4 est, quant à lui, dédié à un problème plus spécifique, où les décideurs souhaitent réduire le nombre de choix possibles. Dans une telle situation, nous utilisons plusieurs règles de dominances non-conventionnelles, autres que la dominance de Pareto, afin d'affiner la sélection. En effet, ces règles de dominances permettent d'établir les préférences parmi les solutions qui sont équivalentes selon la dominance de Pareto. Elles sont ensuite implémentées dans les algorithmes que l'on a développés dans le chapitre précédent. Nous comparons ainsi les méthodes hybridées avec le NSGAII et NSGAIII d'origine afin d'observer les effets apportées par les différentes règles de dominance.

Enfin, le dernier chapitre est consacré à la conclusion générale de ce travail, ses contributions principales, une synthèse de publications, ainsi que les perspectives qu'elle ouvre pour les travaux futurs.

Chapitre 1

État de l'art

Ce premier chapitre est dédié à l'état de l'art pour le problème étudié dans cette thèse – multi-objectif Resource Constraint Project Scheuling Problem (MORCPSP). Une présentation générale du RCPSP est donnée dans la section 1.1. Ensuite, nous nous concentrerons sur la littérature du MORCPSP. Dans la section 1.3, nous faisons une revue de littérature de certaines méthodes de résolution pour le MO-RCPSP. Finalement, dans la section 1.4, plusieurs critères d'évaluation de performance pour les MOOP sont présentés.

1.1 Problème d'ordonnancement sous contraintes de ressources

Les premiers travaux du RCPSP remontent à 1969 par Pritsker et al. [56] avec une formulation en binaire. Les auteurs avaient proposé le premier modèle mathématique du RCPSP. Quelques années plus tard, en 1983, il a été prouvé que le RCPSP est NP-difficle au sens fort par Blazewicz et al. [9]. En 1997, Kolisch and Sprecher ont publié le benchmark le plus utilisé du RCPSP – Le PSPLIB (1997) [36], généré par le logiciel ProGen (1995) [37]. La création de ces instances permet aux chercheurs de se référer à la même base et de comparer leurs résultats sur des instances communes, car, en effet, le RCPSP a attiré de plus en plus d'auteurs différents.

1.1.1 Description du problème

Le "Resource Constraint Project Scheduling Problem (RCPSP)", est avant tout un problème d'ordonnancement de projet. De ce fait, nous considérons un ensemble de n tâches non-interruptibles $J = \{1, 2, \ldots, n\}$, soumises à une série de contraintes de précédence A. Le projet est accompli lorsque toutes les tâches sont terminées – la durée du projet, aussi appelée "le makespan", est caractérisée par la fin de la dernière tâche effectuée. Afin de représenter le début et la fin du projet,

deux tâches fictives sont créées – la tâche 0 pour commencer et la tâche $n+1$ pour conclure ; notons $J' = J \cup \{0, n+1\}$. Dans la Figure 1.2, un exemple du graphe de précédence $W = (J', A)$ avec 7 tâches (dont 5 tâches non-fictives) est donné, où les relations de précédence sont explicitées.

La spécificité principale du RCPSP par rapport à l'ordonnancement de projet est le respect de contraintes liées avec disponibilités des ressources. Lors de son exécution, une tâche peut nécessiter une ou plusieurs quantités ressources pour certaines quantités. Notons l'ensemble de ressources par R ; une ressource $r \in R$ peut être *renouvelable* ou no, elle est renouvelable si sa disponibilité est remise au niveau initial (sa capacité maximale) au début de chaque période durant l'horizon de l'ordonnancement T.

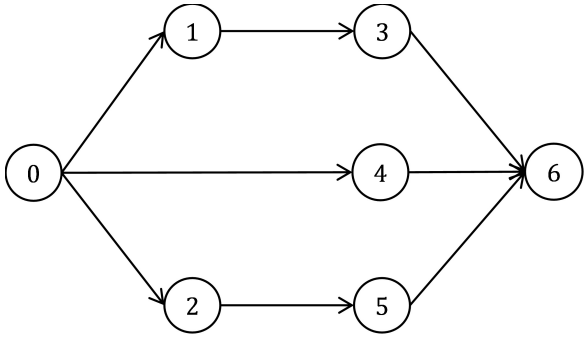

Figure 1.1 – Un exemple de graphe $G = (J', A)$

Tableau 1.1 – Un exemple avec 7 tâches

Job (i)	0	1	2	3	4	5	6
$Pred_i$	-	0	0	1	0	2	3,4,5
p_i	-	2	4	1	3	2	-
r_i	-	4	1	3	2	4	-

Tenant compte de ces deux éléments, nous cherchons à proposer une séquence des tâches, en décidant du début (et éventuellement de la fin) de chaque tâche dans J'. Reprenons l'exemple de la Figure 1.1, nous détaillons les données de cet exemple dans le Tableau 1.1 ; où, une seule ressource est considérée, avec une capacité périodique de 5. Pour chaque tâche $i \in \{0, 1, \ldots, 6\}$, ses prédécesseurs, sa durée d'opératoire et sa consommation de ressource sont donnés respectivement dans les lignes $Pred_i$, p_i et r_i. Si les tâches sont effectuées dans l'ordre 0-1-2-3-4-5-6, nous obtenons la séquence montrée dans la Figure 1.2.

Une tâche $i \in J'$ peut commencer à l'instant $t \in \{0, 1, \ldots, T\}$ si 3 conditions sont satisfaites – les tâches qui se situent devant i dans la séquence sont toutes commencées ; toutes les prédécesseurs de i sont terminées à l'instant t ; il y a suffisamment de ressources à chaque période où la tâche i est à traiter.

Concrètement, la tâche 1 est considérée en premier, elle commence à l'instant 0 comme elle n'a pas de prédécesseur non-fictive et il y a suffisamment de ressources de $t = 0$ à $t = 2$. Pour les mêmes raisons, la tâche 2 débute aussi à $t = 0$. Ensuite, la tâche 3 possède la tâche 1 comme prédécesseur, qui finit à $t = 2$; après avoir vérifié la disponibilité de la ressource, la date de début de la tâche 3 est fixée à $t = 2$. La tâche 4 n'a pas de prédécesseur non-fictif ; en revanche, elle ne peut que commencer après $t = 3$ car la quantité de ressources disponibles n'est pas suffisante. La

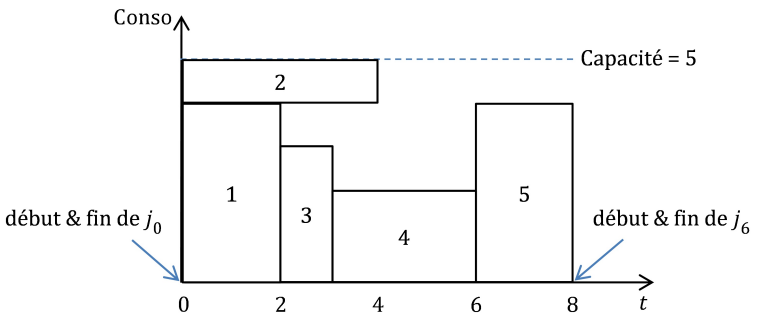

Figure 1.2 – Exemple du planning

date de début de la tâche 5 est conditionnée par la fin de la tâche 2 à $t = 4$, mais aussi par le niveau de ressource qui est suffisant seulement à $t = 6$. Par conséquent, elle commence à $t = 6$. Enfin, la tâche fictive 6 commence et se termine à t = 8, et ceci permet de calculer le makespan = 8.

Ainsi, en plus du makespan, nous pouvons en déduire plusieurs critères pour évaluer la performance du planning : le nombre de tâches en retard, le maximum de l'utilisation des ressrouces au cours de l'horizon, etc.

1.1.2 Classification des problèmes du type RCPSP

Avec l'augmentation du nombre des études du RCPSP, beaucoup de travaux ont été proposés pour challenger ce problème. Brucker et al. (1999) [12] et Hartmann et Briskorn (2010) [28] ont proposé des revues de littérature afin de mieux classer les différentes extensions du RCPSP et de résumer les études apportées à ce sujet. En décrivant précisément les tâches, les ressources, les contraintes du temps et les objectifs, plusieurs variantes du RCPSP ont été identifiés. Dans ce rapport, nous présentons quelques exemples, notamment ceux qui nous paraissent les plus intéressants pour nos travaux.

- RCPSP : Pour conclure la fin du projet, il faut compléter l'ensemble de tâche *non-interruptibles* J. Les tâches doivent respecter l'ensemble de contraintes de précédence W : une tâche peut commencer *à partir du moment* où tous ses prédécesseurs sont activés. Les tâches actives peuvent demander une ou plusieurs *ressources renouvelables* appartenant à l'ensemble de ressources R. L'objectif est de minimiser le *makespan*, la date de fin du projet
- RCPSP avec des variantes portant sur les tâches

  – *Preemption des tâches (P-RCPSP)* : les tâches peuvent être suspendues et reprises ultérieurement, sans aucun impact sur les durées de tâches et demandes des ressources

  – *Multi-mode (MM-RCPSP)* : une tâche peut être exécutée selon plusieurs modes, chacun desquels impliquant éventuellement des durées différentes de tâche et demandes des

ressources

– *Multi-projet (MP-RCPSP)* : plusieurs sous-projets sont considérés en parallèles, la fin du projet pouvant alors être conclue une fois que tous les sous-projets sont terminés

- RCPSP avec des variantes portant sur le temps

  – *Dates d'arrivée des tâches* : toutes les tâches ne sont pas disponibles en même temps, une tâche ne peut commencer qu'après son arrivée

  – *Due dates des tâches* : une date de fin souhaitée est attribuée à chaque tâche. Au-delà de cette date, elle est considérée en retard, ce qui entraîne une éventuelle pénalité

- RCPSP avec des variantes portant sur les ressources

  – *Ressources non-renouvelables* : aussi dite "ressources consommables", où les ressources sont mises en place en une seule fois ou progressivement durant le projet. Contrairement aux ressources renouvelables, leur disponibilités ne sont pas ramenées au maximum au début de chaque période ; d'ailleurs, les quantités de ressources sont cumulables

  – *Capacités de ressources variables* : les disponibilités de ressources varient d'une période à l'autre. Ceci est un cas entre le renouvelable (non cumulable) et le consommable (quantité variante)

- RCPSP avec des variantes portant sur les objectifs

  – *Avance / retard* : établi sur la notion du temps, l'avance et le retard estiment le timing des fins des tâches. Selon les hypothèses dans certaines études, si une tâche est finie avant / après la date de fin souhaitée, cela enchaîne des pénalités que l'on souhaite, dans tous les cas, minimiser.

  – *Valeur actuelle nette (NPV)* : ce critère concerne principalement l'aspect financier. Sachant que l'accomplissement des tâches permet d'encaisser l'argent, et que la consommation de ressource signifie la dépense. La valeur actuelle nette représente la valeur du projet tenant en compte de l'état d'avancement du projet

  – *Coût* : cet objectif dépend souvent des autres critères – le makespan, le retard et l'avance des tâches, etc. La minimisation du coût permet de combiner plusieurs objectifs selon la relation établie.

  – *Multi-objectif (MO-RCPSP)* : plusieurs objectifs *contradictoires* sont considérés, où nous cherchons un ensemble de solutions au lieu d'une seule solution optimale. La recherche des solutions est établie sur la notion de *dominance de Pareto*, qui permet de classer les solutions en *fronts non-dominés*

  – *Latences des tâches* : une durée d'attente est imposée entre la date de début d'une tâche $i \in J$ et un/plusieurs de ses prédécesseurs

Sans développer les détails pour chaque problème, nous présentons dans la figure 1.3 un classement qui permet de résumer les variantes du RCPSP. Les problèmes sont catégorisés selon les

propriétés des tâches, ressources et finalement les critères considérés. Dans cette thèse, nous considérons un RCPSP dans un contexte multi-objectif, aussi dit "MO-RCPSP". Le problème multi-objectif est fréquemment abordé par les industries, qui cherchent à maîtriser le compromis entre plusieurs critères. Nous considérons dans nos études, la minimisation du makespan, la minimisation du retard des tâches et la maximisation de l'équilibrage d'allocation de ressources.

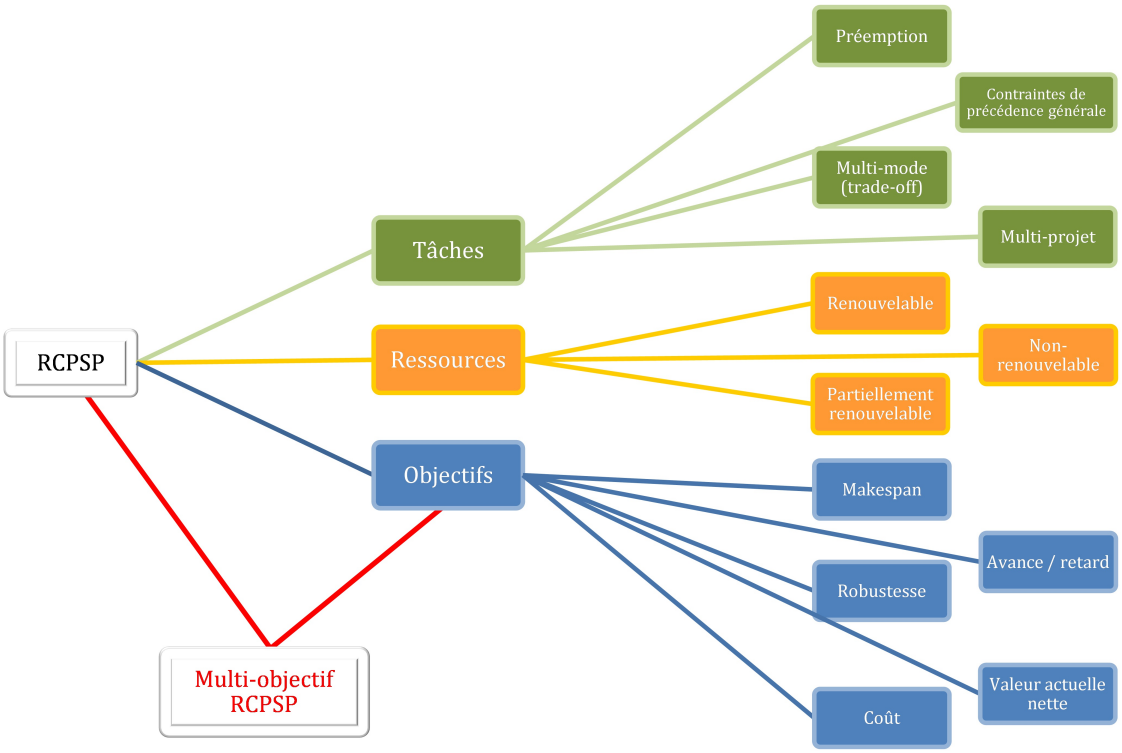

Figure 1.3 – RCPSP : Exemple de classification des problèmes d'extension

1.1.3 Travaux consacrés à la résolution du RCPSP

Cette sous-section est dédié à une présentation sur les travaux concernant le RCPSP et ses variantes. Dans la Figure 1.4, nous fournissons le nombre annuel d'études depuis la première publication en 1969, prenant en compte les 303 études trouvées dans la littérature. Comme on peut le voir sur le graphe, depuis les années 90s, le RCPSP reçoit de plus en plus d'attention ; surtout depuis 2008, beaucoup de travaux ont été développés.

A partir de la base de données que nous avons a constitué, nous avons pu constater l'attention croissante que le RCPSP a reçu durant ces dernières années, mais aussi prendre connaissance plus précisément des sujets qui ont été traités dans la littérature ainsi que les méthodes de résolution utilisées. Dans les sections 1.3.1.1 et 1.3.1.2, nous présenterons les travaux consacrés à la résolution

du RCPSP au sens général. Dans la section 1.2, une revue spécifique de la littérature du MO-RCPSP sera présentée.

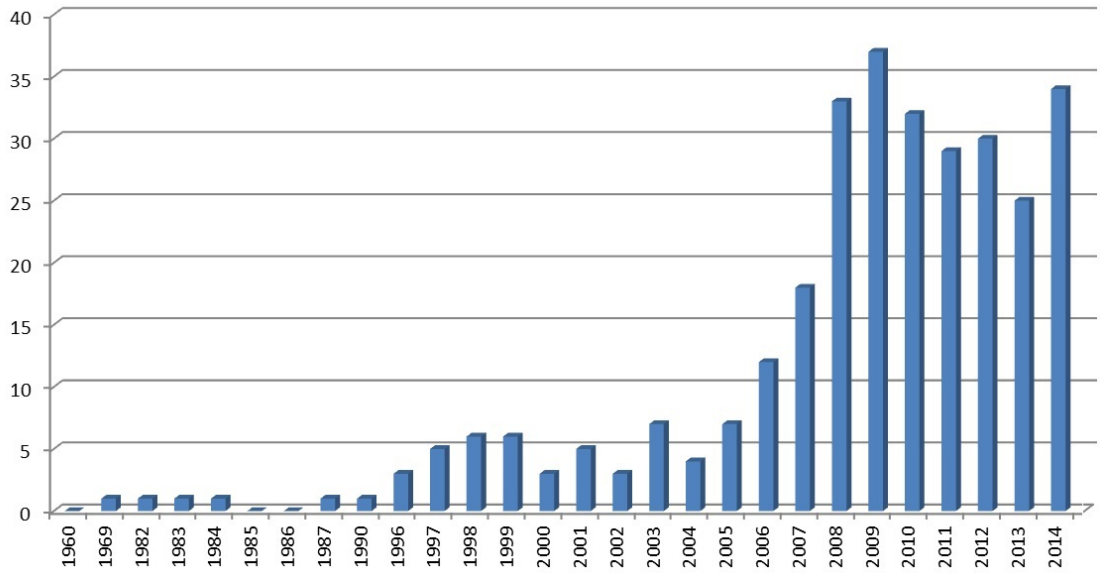

Figure 1.4 – RCPSP : Evolution du nombre de publications trouvées dans la littérature

1.1.3.1 Solutions exactes

Depuis les premiers travaux de Pritsker et al. (1696) [56], les chercheurs mettent l'accent sur la résolution exacte du RCPSP. Les modèles mathématiques sont construits et les Procédures de Séparation et Évaluation (PSE, "Branch and Bound" en anglais) sont aussi mises en place pour trouver les solutions optimales.

Le premier modèle mathématique du RCPSP est présenté par Pritsker et al. (1969) [56]. Pour chaque tâche $i \in J$, chaque instant t avec $0 \leq t \leq T$ (T l'horizon de l'ordonnancement), une variable binaire x_{it} est définie, qui prend la valeur 1 si la tâche i finit à l'instant t, la date de fin de la tâche i est par conséquent formulée par $C_i = \sum_{t=0}^{T} t x_{it}$. Ainsi, nous pouvons en déduire les critères généraux dans les problèmes d'ordonnancement – par exemple, le makespan du projet, comme étant la date de fin de la tâche fictive $n+1$, peut être écrit par $C_{max} = \sum_{t=0}^{T} t x_{(n+1)t}$. Cette formulation est usuelle dans les problèmes d'ordonnancement.

Un autre modèle très visible dans la littérature est proposé par Mingozzi et al. (1997) [45], ayant pour l'objectif de minimiser le makespan. Il permet de calculer une borne inférieure de manière très efficace. Mise à part les variables x_{it} comme dans le modèle de Pritsker et al. (1969) [56], l'originalité de ce modèle est de créer des ensembles de tâches compatibles, notés par $S_h, h = 1, \ldots, H$ (H le nombre d'ensembles valides). Autrement dit, chaque S_h contient des solutions

qui peuvent être exécutées en parallèle, tout en respectant les contraintes de précédence et de ressources. De plus, pour chaque période t, $t = 1, \ldots, T$, un ensemble de tâches compatibles S_{ht} est choisi afin d'expliciter les tâches encours. Dans ce but, une variable binaire p_{ht} est définie pour chaque période, sachant qu'elle prend la valeur 1 si l'ensemble $S_h, h = 1, \ldots, H$ est choisi pendant la période t, $t = 1, \ldots, T$. Ce modèle est particulièrement performant lorsque le nombre d'ensembles compatibles est faible. En revanche, son efficacité décroit grandement lorsque ce nombre est élevé.

Le modèle de Klein (2000) [34] est un des modèles les plus performants. Ce modèle considère les variables "d'état" s_{it} et f_{it}, où s_{it} et f_{it} prennent la valeur 1 si la tâche $i \in J'$ a déjà commencé et respectivement terminé lors de la période t, $t = 1, \ldots, T$, et 0 sinon. Il convient de noter que x_{it} et y_{it} sont des variables non-décroissantes au cours du temps, car si une tâche a déjà commencé (terminé) à certaine période, elle ne peut plus revenir à l'état non-commencé (respectivement non-terminé). Par conséquent, nous pouvons écrire pour toute tâche $i \in J$ sa date de fin $C_i = T - \sum_{t=0}^{T} f_{it} + 1$, et éventuellement le makespan $C_{max} = C_{n+1} = T - \sum_{t=0}^{T} f_{(n+1)t} + 1$. Nous pouvons aussi identifier pour chaque période t les tâches encours, celles qui ont déjà commencé mais pas encore terminé, où $f_{it} - s_{it} = 1$. La consommation de ressource $r \in R$ durant $t = 1, \ldots, T$ peut donc être écrite par $\sum_{i \in J}(f_{it} - s_{it}) * q_{ir}$, ce qui ne doit pas dépasser la capacité périodique de r.

Basé sur le modèle de Klein (2000) [34], Bianco et Camaria (2013) [8] ont également utilisé les variables d'état – "commencer" et "terminer" (avec les mêmes notations s_{it} et f_{it}, comme dans le modèle de Klein (2000) [34]). D'ailleurs, un niveau d'accomplissement est calculé pour chaque tâche pendant chaque période, noté par $y_{it}, i \in J, t \in 1, \ldots, T$. Au début, toutes les tâches sont complétées à 0%. Lors qu'une tâche i est sélectionnée, y_{it} commence à évoluer. Pendant chaque période où la tâche est active, y_{it} augmente d'une quantité de $\frac{1}{p_i}$ (p_i étant la durée de la tâche i). Quand y_{it} atteint 100%, la fin de la tâche est conclue. A partir de cette formulation, Bianco et Camaria (2013) [8] ont proposé une borne inférieure en utilisant la relaxation lagrangienne, et ensuite un procédure de Branch-and-Bound (B&B) pour trouver la solution optimale.

Mis à part ceux qui sont citées ci-dessus, les travaux de Demeulemeester et Herroelen (1992) [21], Brucker et al. (1998) [13], Koné et al. (2011) [38] se sont aussi intéressés à trouver la solution exacte du RCPSP : soit avec la méthode B&B, soit en proposant les modèles mathématiques innovants. Ces travaux ont proposé des approches innovantes pour résoudre le RCPSP de manière optimale.

1.1.3.2 Solutions approchées

Dans Blazewicz et al. (1983) [9], il a été démontré que le RCPSP est un problème NP-difficile au sens fort. Ceci indique donc que les méthodes exactes ne permettront que résoudre les problèmes de petites tailles. Par conséquent, beaucoup de chercheurs ont proposé des méthodes approchées afin de résoudre le RCPSP. De plus, contrairement aux résolutions exactes (dont le problème visé est le RCPSP original dans la plupart des cas), quand les méthodes approchées sont impliquées, beaucoup de variantes du problème RCPSP sont considérés.

Dans Baar et al. (1999) [5], les auteurs ont considéré deux procédures du type recherche taboue (*Tabu Search, TS*) pour résoudre le RCPSP en minimisant le makespan. La première méthode est établie sur la notions d'arcs critiques, où un opérateur intervient sur les arcs qui se situent sur le chemin critique. La deuxième procédure intègre une recherche du voisinage, qui s'est établie sur les relations parallèles des tâches. Dans le cadre de la maximisation de la robustesse, Lambrechts et al. (2008) [40] ont considéré une fonction objectif en relation avec le nombre de slacks libres dans le planning et résolu le problème avec la méthode TS.

Les travaux de Pinson et al. (1994) [54], Boctor et al. (1996) [10] font partie des premières études résolvant le RCPSP avec le recuit simulé (*Simulated Annealing, SA*). Mis à part le RCPSP original, les auteurs ont aussi considéré le MM-RCPSP. Le MM-RCPSP est aussi pris en compte par Bouleimen et Lecocq (2003) [11], où les auteurs s'appuyaient sur la spécification de l'espace de solution du RCPSP ; l'efficacité du SA proposé est prouvée par les expériences. Ceci a fait que cette version est parmi les plus puissants algorithmes pour résoudre le RCPSP et le MM-RCPSP. Plus récemment, Mika et al. (2005) [44] ont comparé le recuit simulé et la recherche taboue sur un RCPSP avec la maximisation de valeur actuelle nette. Pour le problème étudié, la recherche taboue est plus performante sur les petites instances alors que le recuit simulé montre ses avantages lors que la taille du problème augmente.

Les algorithmes génétiques (*Genetic algorithms, GA*), en tant qu'une des méta-heuristiques les plus performantes, sont déjà très présents dans la littérature concernant le RCPSP. Les travaux de Hartmann (1998) [29] fait partie des premiers travaux résolvant le RCPSP avec les GAs, où les auteurs ont proposé un chromosome codé par une "liste d'activités". Ce codage crée une séquence, où chaque position est affectée à une tâche $i \in J$. Lors de la construction de la séquence, les tâches sont considérées l'une après l'autre, selon leur position dans la liste d'activités. Toutes les tâches commencent le plus tôt possible, sous réserve d'un niveau suffisant des ressources disponibles. Il a été prouvé par des expériences que ce codage est très efficace et intégré dans plusieurs travaux, tels que Hartmann (2002) [30], Zamani (2013) [84] pour résoudre le RCPSP, Okada et al. (2014) [51] pour le Multi-Mode RCPSP (MM-RCPSP). Plus généralement, les GAs peuvent aussi être trouvés dans les travaux de Vanhoucke (2009) [76] pour maximiser la valeur actuelle nette ; Van Peteghem

et Vanhoucke (2010) [74] ont utilisé un bi-population GA pour résoudre un MM-P-RCPSP (multi-modes et tâches interruptibles) en minimisant le makespan ; un GA avec des clés aléatoires est appliqué dans les travaux de Mendes et al. (2009) [43] pour résoudre le RCPSP original.

Une autre méthode méta-heuristique fréquemment implémentée dans la littérature du RCPSP est l'optimisation par essaims particulaires (*Particle Swarm Optimization, PSO*). Dans les travaux de Zhang et al. (2005) [85], deux stratégies sont considérées afin de construire les particules – une basée sur les pondérations de priorités et l'autre sous forme de liste d'activités. Les pondérations de priorités sont calculées en tenant en compte de plusieurs paramètres, dépendant des caractéristiques des données. Ceci permet d'affecter un poids à chaque tâches afin de décider de leur position dans la séquence d'ordonnancement. Quant à la liste d'activités, les auteurs ont rejoint les travaux de Hartmann (1998) [29]. Il est reconnu par des expériences, que la représentation avec la liste d'activités est plus performance celle avec les pondérations de priorités. D'ailleurs, l'efficacité de la PSO proposée est aussi montrée par une étude comparative avec un algorithme génétique, un recuit simulé et une recherche taboue.

Dans ce document, nous avons mis en avant les méta-heuristiques en tant que méthodes approchées. Il faut aussi noter que plusieurs heuristiques intéressantes ont été développées pour le RCPSP. Par exemple, dans un rapport technique de Kolisch et Hartmann (1998) [35], les auteurs ont comparé plusieurs heuristiques basées sur les règles de priorité (*Shortest Processing Time*, *Minimum Slack*, etc.) Ils ont également considéré une méthode qui procède alternativement à l'avance et au recul lors de la construction de solution. L'idée d'alterner entre l'avance et le recul peut aussi être trouvée dans les travaux de Valls et al. (2004) [72], où les auteurs ont couplé une telle procédure avec un *Convex Search Algorithm* afin d'obtenir un ensemble de solutions de haute qualité.

1.2 RCPSP avec multiples objectifs

Dans la section précédente, nous avons fait une synthèse sur la littérature du RCPSP au sens général, surtout concernant les problèmes mono-critère. Comme décrit dans le chapitre d'introduction, l'intérêt pour les ressources est devenu de plus en plus important pour les industries, où les cas mono-objectif est rarement représentatif de la réalité. C'est pourquoi nous choisissons dans nos études, un RCPSP avec multiples objectifs.

1.2.1 Solutions approchées

En 1981, Slowinski (1981) [67] a introduit les études multi-objectifs dans le contexte du RCPSP. Tenant en compte du makespan, du retard maximal et des coûts, le problème multi-objectif

est résolu par une programmation linéaire multi-objectif. Plusieurs cas de figure considérés – tâches interruptibles, multi-modes, ressource renouvelable, non-renouvelable et partiellement renouvelables. Quelques années après, Slowinski et al. (1994) [68] ont proposé un système d'aide à la décision, qui a fait référence à plusieurs travaux de ce même auteur Slowinski (1989) [65], Slowinski et al. (1991) [66]. Ce premier s'est focalisé sur un RCPSP avec les tâches non-interruptibles, multi-modes, avec les ressource renouvelable, non-renouvelables et enfin partiellement renouvelables. Plusieurs critères sont pris en compte, y compris le makespan, le retard total pondéré, le nombre de tâches en retard, le lissage de l'utilisation des ressources, etc. Le dernier objectif mentionné pourrait être similaire à un des critères considérés dans nos travaux (équilibrage d'allocation des ressources). Cependant, les explications concernant ce critère sont incomplètes dans la publication de Slowinski (1994) [68]. Les auteurs ont abordé 3 méthodes de résolution, une heuristique établie sur les règles de dominances, une méta-heuristique (le recuit simulé) et un branch-and-bound. L'objectif de cette étude est de développer un outil appliqué sur les vrais instances, pour cette raison, les auteurs n'ont pas forcément exigé sur l'exactitude des solutions, mais plutôt limité le temps de calcul.

En ce qui concerne les travaux plus récents, les chercheurs sont plutôt intéressés par les problèmes avec moins de critères (principalement 2 ou 3 objectifs). Les auteurs Viana et De Sousa (2000) [78] ont résolu un MO-RCPSP avec la minimisation du makespan, du retard total pondérés des tâches et de la violation des contraintes de ressources. Le recuit simulé Pareto (*Pareto Simulated Annealing, PSA*) et une recherche taboue multi-objectif (*Multi-Objectif Tabu Search, MOTS*) sont mis en œuvre pour résoudre le problème de façon approchée.

La minimisation du makespan et la maximisation de la robustesse sont considérés dans les études de Al-Fawzan et Haouari (2005) [2] et Abbasi et al. (2006) [1]. La robustesse, en tant que critère en relation avec la flexibilité du planning, est modélisée par le nombre de slacks libres (aussi appelé "temps flottant"). Pour résoudre le problème, Al-Fawzan et Haouari (2005) [2] ont utilisé un MOTS, où la recherche taboue (TS) est lancée plusieurs fois avec des fonctions d'agrégation différentes pour trouver plusieurs solutions non-dominées. Abbasi et al. (2006) [1], quant à eux, ont proposé une fonction objectif linéairement dépendante du makespan et de la robustesse. Un algorithme SA est utilisé, avec un procédure de récursion "forward-backward" pour la recherche du voisinage. Certaines contraintes virtuelles sont ajoutées au fur et à mesurer afin de trouver plusieurs solutions.

Les algorithmes génétiques sont également utilisés pour résoudre les MO-RCPSP. Khalili et al. (2013) [33] ont considéré un bi-objectif RCPSP avec la minimisation du makespan et la maximisation de la valeur actuelle nette. Les auteurs ont utilisé deux algorithmes génétiques avec multi-population pour résoudre le problème. Vanucci et al. (2012) [77] ont proposé une version modi-

fiée de “Non-dominated Sorting Genetic Algorithm" (NSGAII), qui s’appuie spécialement sur le choix des solutions faisables, pour minimiser le makespan et le coût total dans un MM-RCPSP. Ce même problème est aussi inclut dans les études de Ghoddousi et al. (2013) [24], où la variation des consommations des ressources sont aussi prises en compte par l’intermédiaire un troisième critère. Le NSGAII est à nouveau implémenté pour résoudre le problème à 3 objectifs. Dans les travaux de Kazemi et Tavakkoli-Moghaddam (2011) [32], les opérateurs des algorithmes génétiques sont couplés avec la méthode PSO. Sur un problème bi-objectif avec la minimisation du makespan et la maximisation de la valeur actuelle nette, la méthode hybridée est reconnue plus robuste que le NSGAII.

Dans nos études, nous avons implémenté le NSGAII et le NSGAIII pour résoudre un MO-RCPSP avec 3 objectifs Wang , Yalaoui et Dugardin [80]. En plus, nous avons proposé des améliorations, des techniques plus appropriées vis-à-vis des spécifications de notre problème [80], [83], [82].

1.2.2 Solutions exactes

Sachant que tous les travaux cités ci-dessus concernant la résolution approchée, peu de travaux ont été consacrés à résoudre le MO-RCPSP de manière exacte. Gutjahr (2015) [26] a considéré le compromis entre le makespan et le coût pour MM-RCPSP avec des durées et des coûts de tâches stochastiques. Un algorithme de branch-and-bound est mis en place pour trouver le front de Pareto optimal. Dans nos études Wang, Dugardin et Yalaoui (2016) [79], nous avons résolu un bi-objectif RCPSP avec la minimisation du makespan et le retard total des tâches. Nous nous sommes inspirés du modèle mathématique de Klein (2000) [34] et proposé un modèle bi-objectif. La méthode exacte à deux phase (TPM) est ensuite mise en œuvre pour résoudre le problème de manière exacte.

1.3 Méthodes de résolutions générales pour les problèmes multi-objectif et RCPSP

Dans cette section, nous mettons l’accent sur les méthodes générales pour les problèmes multi-objectif. Dans un premier temps, on s’appuie sur certaines méthodes exactes afin de trouver le front de Pareto optimal. Ensuite, nous présenterons les méthodes approchées fréquemment sollicitées dans la littérature du MO-RCPSP.

1.3.1 Quelques notions importantes des problèmes multi-objectif

1.3.1.1 Formulation générale

Les problèmes multi-objectif considèrent plusieurs critères contradictoires. Prenons l'exemple d'un problème de minimisation avec K objectifs $\mathcal{P}$, il peut être défini comme ci-dessous :

$$\text{Minimiser} \quad F(X) = \left(f_1(X), f_2(X), \ldots, f_K(X)\right)^T \tag{1.1}$$

$$\text{Sous contraintes} \quad g_j(X) \leq 0, \quad \forall j \in \{1, 2, \ldots, m\} \tag{1.2}$$

$$l_i \leq x_i \leq u_i, \quad \forall i \in \{1, 2, \ldots, n\} \tag{1.3}$$

où X est le vecteur de variables de décision $X = (x_1, x_2, \ldots, x_n)$ (n le nombre de variables) ; $g_j(X)$ sont les contraintes (m étant le nombre de contraintes) ; l_i et u_i représentent la borne inférieure et la borne supérieure de la variable x_i, respectivement.

Pour faciliter la notation, pour une solution s, on note également ses fonctions objectifs comme suit :

$$F(s) = \left(f_1(s), f_2(s), \ldots, f_K(s)\right)^T \tag{1.4}$$

où $f_k(s)$ ($k \in \{1, 2, \ldots, K\}$) est la valeur sur l'objectif k obtenue avec s.

1.3.1.2 Dominance de Pareto

La *dominance de Pareto* est une des clés dans les problèmes d'optimisation multi-objectif. Elle permet d'évaluer et classifier les solutions et proposer un *front de Pareto* en tant que l'ensemble de meilleures solutions.

Soit un problème de minimisation $\mathcal{P}$ avec K objectifs, s_1, s_2 deux solutions valides. Pour une solution $s_n, n \in \{1, 2\}$, notons $f_k(s_n)$ la valeur objectif de s_n sur la critère $k \in \{1, 2, ..., K\}$. La dominance de Pareto est établie si

$$\begin{cases} f_k(s_1) \leq f_k(s_2), \quad \forall k \in \{1, 2, \ldots, K\} & (1.5) \\ \exists k' \in \{1, 2, ..., K\}, \quad f_{k'}(s_1) < f_{k'}(s_2) & (1.6) \end{cases}$$

alors s_2 est dit Pareto-dominée par s_1, noté par $s_1 \prec_P s_2$. Par contre, si $s_1 \not\prec_P s_2$ et $s_2 \not\prec_P s_1$, alors les deux solutions sont dites "non-dominées". Nous pouvons ensuite définir un *front non-dominé*, aussi dit *front de Pareto*, constitué un ensemble de solutions mutuellement non-dominées.

1.3.1.3 Solutions efficaces et supportées

Lors de la recherche des solutions, plusieurs fronts de Pareto peuvent être identifiés, parmi lesquels, nous trouverons un ensemble de solutions *efficaces*. Une solution s est dite efficace s'il

n'existe aucune solution s', telle que $s' \prec_P s$ dans l'espace des solutions. Par conséquent, résoudre les problèmes multi-objectif de manière exacte revient à la recherche des solutions efficaces. Si toutes les solutions efficaces sont trouvées, nous obtenons le *front de Pareto optimal*.

Par ailleurs, lors qu'une programmation linéaire est appliquée, il est nécessaire de distinguer deux types de solutions efficaces – les solutions *supportées* et les *non-supportées*. Une solution est supportée si elle permet d'optimiser le problème pondéré $\mathcal{P}_\lambda$, tel que :

$$\text{Minimiser} \quad F_\lambda(X) = \lambda_1 f_1(X) + \lambda_2 f_2(X) + \ldots + \lambda_K f_K(X) \tag{1.7}$$

$$\text{Sous contraintes} \quad g_j(X) \leq 0, \quad \forall j \in \{1, 2, \ldots, m\} \tag{1.8}$$

$$l_i \leq x_i \leq u_i, \quad \forall i \in \{1, 2, \ldots, n\} \tag{1.9}$$

Les équations (1.8) et (1.9) correspondent aux équations (1.2) et (1.3), respectivement, dans la section 1.3.1.1. L'équation (1.7) donne la fonction objectif pondérée, qui est linéairement dépendante des critères $f_k, k = 1, 2, \ldots, K$; où λ_k est désigné au poids affecté à l'objectif $k = 1, 2, \ldots, K$.

Au contraire, si une solution efficace ne peut pas être trouvée de cette façon, elle est non-supportée.

1.3.2 Méthodes exactes

Afin de trouver toutes les solutions efficaces dans un problème d'optimisation multi-objectif (MOOP), plusieurs méthodes ont été proposées dans la littérature. Le Branch-and-Bound, la programmation dynamique sont aussi connus dans le domaine multi-objectif que le contexte mono-critère. Dans ce rapport, nous présentons 3 approches développées pour les problèmes multi-objectif : la méthode ε-contrainte, la méthode à deux phase et la méthode de partitionnement parallèle.

1.3.2.1 Méthode ε-contrainte

Introduite par Haimes et al. (1971) [27], la méthode ε-contrainte est une des méthodes les plus référencées pour la solution exacte des problèmes multi-objectif. Pour un problème multi-objectif $\mathcal{P}$ avec K objectifs à minimiser, un seul objectif $f_{k'}$ ($k' \in \{1, 2, \ldots, K\}$) est considéré comme critère principal alors que les autres objectifs $k \neq k' (k \in \{1, 2, \ldots, K\})$ sont transformés sous

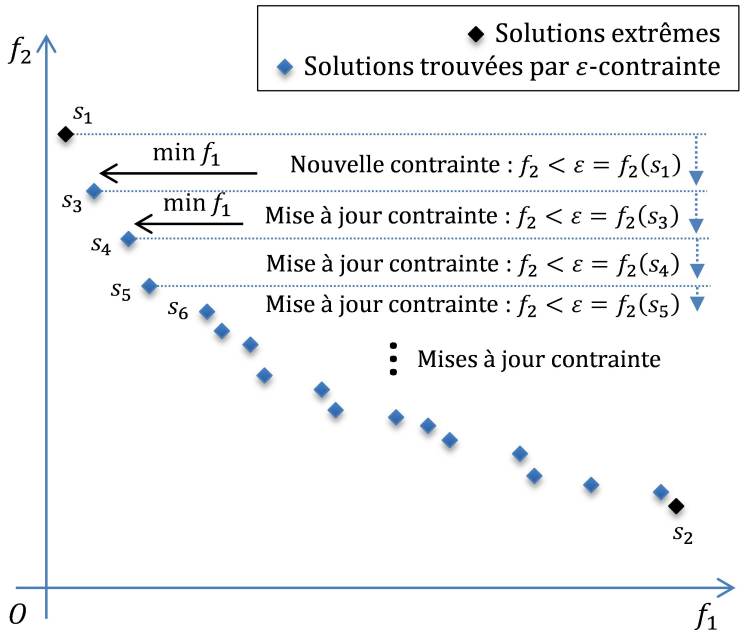

Figure 1.5 – Exemple d'un front de Pareto obtenu avec la méthode ε-contrainte

forme de contraintes.

$$\text{Minimiser} \qquad f_{k'}(X), \quad k' \in \{1, 2, \ldots, K\} \tag{1.10}$$

$$\text{Sous contraintes} \qquad g_j(X) \leq 0, \quad \forall j \in \{1, 2, \ldots, m\} \tag{1.11}$$

$$l_i \leq x_i \leq u_i, \quad \forall i \in \{1, 2, \ldots, n\} \tag{1.12}$$

$$f_j(X) < \varepsilon_j, \quad \forall k \neq k' | k \in \{1, 2, \ldots, K\} \tag{1.13}$$

L'objectif choisi est explicité dans l'équation (1.10) ; les équations (1.8) et (1.9) correspondent aux équations (1.2) et (1.3) dans la section 1.3.1.1 ; enfin, les contraintes spéciales de la méthode sont transcrites par l'équation (1.13).

Pour montrer le déroulement de la méthode, nous donnons un exemple bi-objectif comme montré dans la Figure 1.5. Pour commencer, les deux problèmes mono-objectifs sont résolus. Nous obtenons deux solutions extrêmes, notée par s_1 et s_2. f_1 est ensuite pris comme le critère principal et f_2 s'est transformé en contrainte. Concrètement, dans un premier temps, nous ajoutons la contrainte $f_2 < \varepsilon = f_2(s_1)$ et continuons à minimiser f_1. Notons la solution trouvée par s_3, nous mettons à jour ensuite la valeur de ε, tel que $\varepsilon = f_2(s_3)$, ainsi $f_2 < f_2(s_3)$. Avec la même démarche, nous pouvons trouver s_4, s_5, s_6, ..., et finalement toutes les solutions efficaces, qu'elle soient supportées ou non (définition voir section 1.3.1).

Dans l'exemple donné, la méthode ε-contrainte résout une série de problèmes mono-objectifs sur f_1, en limitant les valeurs sur f_2. Plus généralement, pour un problème avec K objectifs, la méthode vise toujours un seul critère. Le nombre de dimensions est en réalité baissé 1 par 1 – pour

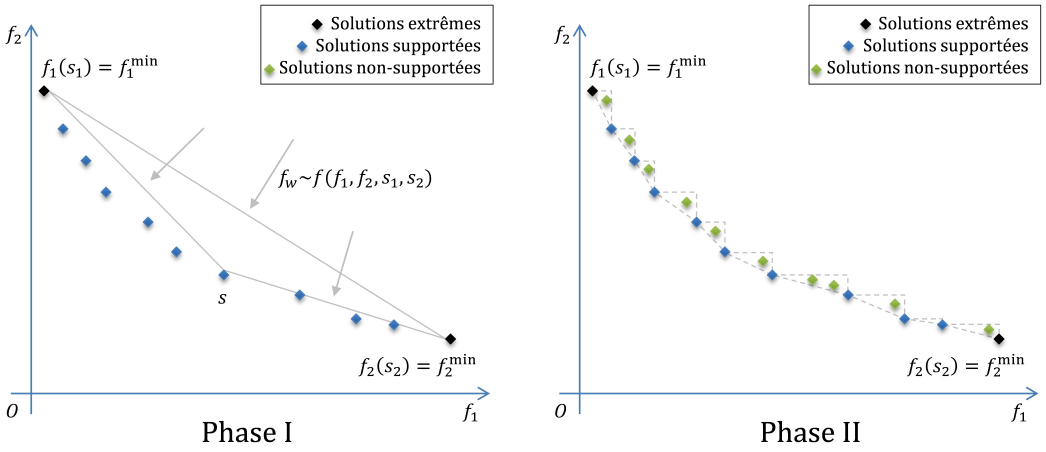

Figure 1.6 – Exemple d'un front de Pareto obtenu avec la TPM

mettre les autres $K-1$ objectifs en contrainte, nous devons considérer $K-1$ sous-problèmes, chacun parmi lesquels nécessite de transformer $K-2$ critères en contrainte... et ainsi de suite. De ce fait, un inconvénient de la méthode ϵ-contrainte est que le temps de calcul croît assez rapidement quand le nombre d'objectifs augmente.

1.3.2.2 Méthode à deux phases

La méthode à deux phases (*Two Phases Method, TPM*) est proposée par Ulungu et Teghem (1995) [71], dans le cadre de la résolution d'un problème de sac à dos avec deux objectifs.

La démarche générale de la TPM est de résoudre une série de problème mono-critère avec les objectifs pondérés. La recherche du front de Pareto optimal est faite en deux temps. Pendant la phase I, une série de problèmes pondérés sont résolue et les solutions *supportées* sont obtenues (définition voir section 1.3.1). Ensuite, la phase II consiste à explorer les solutions *non-supportées*, qui se situent dans les zones non-dominées des solutions déjà trouvées. Dans la Figure 1.6, nous donnons un exemple de front de Pareto, distinguant les solutions supportées et non-supportées.

Après avoir trouvé les deux solutions extrêmes s_1 et s_2, un objectif pondéré f_λ, qui dépend de f_1, f_2, s_1 et s_2, peut être décidé. En optimisant cet objectif f_λ, nous obtenons une solution *supportée* – s. Ensuite, deux problèmes sont générés issus de s – en considérant $[s, s_1]$ et $[s, s_2]$, comme montré dans la Figure 1.6. Cette procédure est répétée à chaque fois qu'une nouvelle solution est obtenue. La phase I se termine quand toutes les solutions supportées sont trouvées, c'est-à-dire, toutes les problèmes pondérés sont résolus de manière exacte. Par la suite, les solutions qui ne peuvent être trouvées lors de la phase I sont recherchées. Pour ce faire, une série de problèmes pondérés, tout en excluant les solutions déjà trouvées (que ça soit dans la phase I ou la phase II) sont considérés. Pour plus de détails, nous présenterons les démarches de manière formelle dans

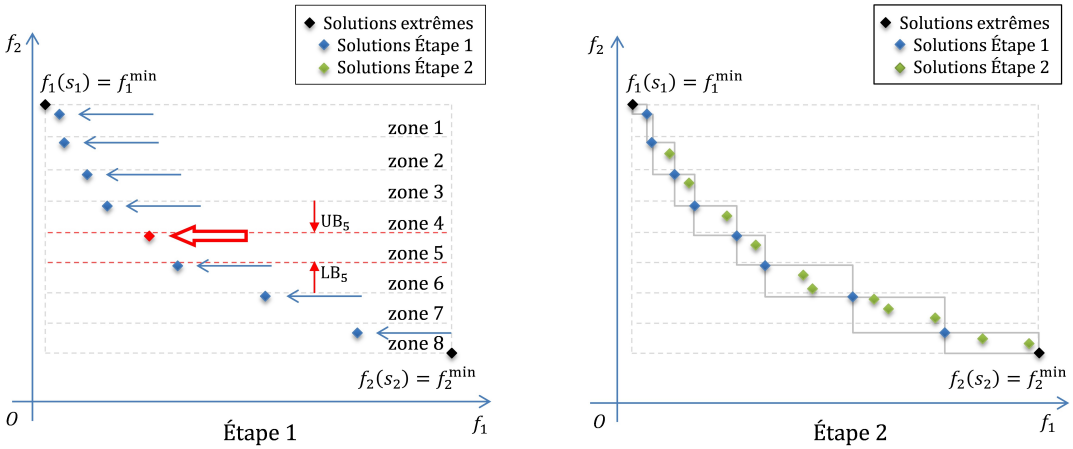

Figure 1.7 – Exemple d'un front de Pareto obtenu avec la PPM

la section 2.

La difficulté de la TPM se trouve à l'exploitation de l'espace de solutions. Chaque nouvelle solution peut entraîner de nouvelles bornes sur toutes les objectifs. Przybylski et al. (2010) [58] ont récemment étendu la TPM à une version multi-objectif, en prenant pour exemple un problème d'affection à 3 critères. Grâce aux techniques de réductions de l'espace de solutions dans ce travail, quand le nombre d'objectifs augmente, le temps de calcul avec la TPM croit, mais pas de manière dramatique comme avec la méthode ε-contrainte.

Dans la littérature, la TPM a été appliquée dans plusieurs contextes afin de résoudre les problèmes bi-objectifs. En s'appuyant sur la performance des algorithmes réorganisés, Przybylski et al. (2008) [57] a utilisé la TPM sur le problème d'affection et testé sa performance sur plusieurs benchmarks. Prins et al. (2006) [55] ont combiné la TPM avec la relaxation lagrangienne pour résoudre un problème bi-objectif de couverture par ensembles (*Bi-objectif Set Covering Problem*). Dans cette étude, la relaxation lagrangienne est appliquée en tant que procédure consécutive lors de la recherche de solutions.

En ce qui concerne le MO-RCPSP, peu de travaux ont été consacrés à la résolution exacte. Dans nos études, nous adopterons la TPM afin de résoudre un bi-objectif RCPSP [79], où la minimisation du makespan et du retard total sont prises en compte.

1.3.2.3 Méthode de partitionnement parallèle

Dans notre travail, nous faisons référence à une autre approche exacte – la méthode de partitionnement parallèle (*Parallel Partionning Method, PPM*), proposée par Lemesre et al. (2007) [41]. L'idée principale de cette méthode est de diviser l'espace de solution en plusieurs parties, de façon parallèle ; ensuite de chercher à optimiser le problème en considérant un seul objectif.

Nous décrivons les démarches principales de la PPM, avec l'exemple du problème bi-objectif de minimisation. Lorsque l'on a deux critères à optimiser, la TPM se déroule en deux temps. Afin de les distinguer par rapport à la TPM, ces phases sont notées “étape 1" et “étape 2". La PPM prend en considération les principes de la méthode ε-contrainte et de la TPM. Lors de l'étape 1, la PPM se focalise sur un seul objectif et transforme les autres critères en contraintes. La scalarisation est considérée dans l'étape 2 pour compléter l'exploration de l'espace de solution ; cette technique est aussi utilisée dans la TPM.

La procédure commence toujours par borner l'espace de solution par s_1 et s_2, en résolvant les deux problèmes mono-critère. Ensuite, lors de l'étape 1, la PPM découpe l'intervalle de solution d'un objectif en plusieurs parties de même taille. Nous illustrons un exemple dans la Figure 1.7, où f_2 est divisé en plusieurs intervalles. En effet, le nombre d'intervalles est un paramètre utilisé dans la PPM et il peut agir sur l'efficacité de la méthode. Dans l'exemple donné, nous avons considéré ce paramètre comme 8 mais il tout de même possible de utiliser une autre valeur. Dans chaque zone, la valeur objectif sur f_2 est bornée par une borne supérieure (notée “UB") et une borne inférieure (notée “LB"). Dans chaque zone, le problème est résolu en minimisant f_1. Par exemple, s est trouvée en optimisant f_1 dans la zone 5, en prenant considération de UB_5 et LB_5. La deuxième étape consiste à explorer les régions non-dominées des solutions trouvées lors de l'étape 1. Pour ce faire, une série de problèmes avec des objectifs pondérés sont résolus. Nous présenterons plus de détails techniques dans la section 2.3.2.

Cette méthode est ensuite généralisée par Dhaenens et al. (2010) [22] pour résoudre les problèmes avec plus de deux critères, appelée “k-PPM" (“k” étant le nombre d'objectifs). L'idée de base de la PPM est reprise, k-PPM s'appuie sur l'exploration de l'espace de solution lorsque plusieurs objectifs sont impliqués. Récemment, dans les travaux de Gomes et al. (2015) [17], les auteurs se sont inspirés de la PPM pour résoudre un bi-objectif problème d'arbre couvrant de diamètre minimal (*Bi-objectif Minimum Diameter-Cost Spanning Tree Problem*).

Malgré le fait que la PPM et la k-PPM soient citées par plusieurs travaux, elles n'ont pas encore bien intégrés dans la littérature des MOOPs. Dans nos études, nous mettrons en place la PPM et la comparerons avec les autres méthodes exactes.

1.3.3 Méthodes approchées

Cette section est consacrée à une présentation sur les méthodes approchées pour résoudre le MO-RCPSP ou, plus généralement, les problèmes multi-objectif. Nous nous appuyons sur les méta-heuristiques. Dans ce rapport, nous avons choisi les algorithmes génétiques et le recuit simulé comme exemples de ce type de méthodes qui permettent de trouver les fronts de Pareto

approchés.

1.3.3.1 Les algorithmes génétiques – NSGAII, NSGAIII

Les algorithmes génétiques (AGs) sont des méthodes évolutionnaires à population. Introduit par Goldberg (1989) [25], les AGs ont connu beaucoup de succès sur les problèmes mono-critères ainsi que les problèmes multi-critères.

Les principales étapes de cette méthode est comme ci-dessous : d'abord, une population initiale est générée comme le point de départ ; les fonctions objectifs de chaque individus sont calculées pour les évaluer. Par la suite, les opérateurs de croisement et mutation interviennent pour créer de nouvelles solutions. Le procédure de sélection interviendra alors afin de choisir la prochaine génération. Ces individus choisis constituent ensuite la génération d'après. De ce fait, la population évolue progressivement, une génération après une autre.

Le "Non-dominated Sorting Genetic Algorithm II" (NSGAII) Deb et al. (2002) [19] est un algorithme génétique développé spécialement pour les problèmes multi-objectif. La spécificité du NSGAII par rapport aux autres AGs se trouve dans la procédure de sélection. Le NSGAII propose un processus basé sur la dominance de Pareto : plusieurs fronts non-dominés sont établis, les solutions sur les meilleurs fronts sont choisies en priori. Ensuite, pour gérer les solutions Pareto-équivalentes, le NSGAII mesure la distribution de solutions dans l'espace de recherche par le "crowding distance".

Comme une version étendue de NSGAII, le NSGAIII Deb et Jain (2014) [18] est conçu pour résoudre les problèmes avec 3 objectifs ou plus. La différence principale entre les deux NSGAs est la façon de distinguer les solutions Pareto-équivalentes. Au lieu d'utiliser le "crowding distance", le NSGAIII crée un hyperplan où se trouve un ensemble de points de référence. Chaque solution est ensuite associée à un unique point de référence. Ce couplage permet donc d'évaluer la distribution des solutions.

1.3.4 Le recuit simulé – PSA

Le recuit simulé Van Laarhoven et Aarts (1987) [73] est une méta-heuristique inspirée d'un processus utilisé en métallurgie. L'idée de cette méthode est de simuler numériquement une opération de traitement thermique de type recuit. On alterne des cycles de refroidissement lent et de réchauffage (recuit) qui ont pour effet de minimiser l'énergie du matériau. En optimisation, cette méthode est transposée pour trouver les extrema d'une fonction.

Le déroulement du recuit simulé (*Simulated Annealing, SA*) nécessite plusieurs paramètres,

dont la température initiale (point de départ, valeur élevée) et la température d'arrêt du recuit (point d'arrêt, valeur basse). Avec une solution de départ, son voisinage est exploité et une nouvelle solution est trouvée. Si la nouvelle solution est meilleure, elle est acceptée. Sinon, elle peut être acceptée avec une probabilité en fonction décroissante de $\frac{\Delta}{T_s}$, où Δ est l'écart entre les deux solution et T_s la température actuelle du système. Au début de la procédure, la température est élevée, alors la probabilité que le résultat soit "dégradé" est relativement grande. Pendant le déroulement de l'algorithme, la température du système décroit progressivement. Par conséquent, cette probabilité devient plus petite, ce qui permet au système de converger vers un état stable.

Dans la littérature, plusieurs versions multi-objectif du recuit simulé sont proposées, comme dans Czyzzak et Jaszkiewicz (1998) [16], Suman (2002) [69], Bandyopadhyay et al. (2008) [7]. Nous prenons l'exemple de la version de Czyzzak et Jaszkiewicz (1998) [16] dans ce rapport, appelé "Pareto Simulated Annealing (PSA)". Dans la version multi-objectif, la plupart de démarches du SA sont conservées. La différence principale est, pour décrire l'état du système, on utilise un ensemble de solutions S au lieu d'une seule s. De plus, le front non-dominé est établi à partir de S. Lors de chaque itération, les voisinages de toutes les solutions $s \in S$ sont explorés. Si la nouvelle solution trouvée n'est pas dominée par s, elle est acceptée. Sinon cette première est acceptée avec une probabilité, qui dépend de la température actuelle, l'ensemble S, la solution s et la nouvelle solution elle-même.

1.4 Évaluation de la performance

Dans la littérature, de nombreuse de travaux sont consacrés à l'évaluation les fronts de Pareto approchés. Mise à part les mesures intuitives comme le temps calcul et le nombre de solutions, les fronts non-dominés sont aussi évalués selon leurs convergences, diversités, etc. Nous citons les publications de Zitzler et al. (2003) [86], Okabe et al. (2003) [50], Jiang et al. (2014) [31], où les métriques ont été analysées en détail. Dans ce rapport, nous présentons plusieurs exemples non-exhaustifs, parmi lesquels, certains seront sollicités pour évaluer les résultats dans les chapitres suivants. Notons PF^* et S le front de Pareto optimal et celui trouvé lors de la résolution, nous avons les métriques qui évaluent S sur :

- La *convergence*
  - Le *ratio d'erreur* (ER), représente la proportion de solutions dans S qui ne sont pas sur le front de Pareto exact PF^*
  - La *distance générationnelle* (GD), proposé par Veldhuizen et al. [75] mesure le niveau d'approximation de S vers PF^* avec les distances Euclidiennes
  - La *distance générationnelle inversée* (IGD) de Sierra et Coello Coello (2005) [64]

est une version modifiée de la GD. Au lieu de mesurer la distance de S à PF^*, l'IGD se base sur les distances Euclidiennes de FP^* à S

- La *divergence*

  – L'*espacement* (SP) de Schott (1995) [63], permet d'évaluer le degré d'uniformité de la distribution des solutions sur S

  – L'*écart maximum* (MS), proposé par Zitzler et al. (2000) [87] calcule la distance diagonale de l'espace bornée par les solutions extrêmes dans S

  – La *métrique de couverture* (C), de Zitzler et al. (2003) [86] compare deux ensembles de solutions, en calculant pour chacun, la proportion de solutions dominées par l'autre

- Les *performances combinées*

  – La *métrique-*ϵ Zitzler et al. (2003) [86] se base sur la ϵ dominance. Elle est calculée par le facteur ϵ qui permet d'établir une relation ϵ-dominée entre deux ensembles de solutions

  – Le *hypervolume* proposé par Zitzler et al. [86] mesure, en prédéfinissant un ensemble de solutions référentielles, l'espace de solution dominée par le front de Pareto concerné

Dans nos études, nous prenons en compte une métrique de chaque catégorie pour l'évaluation des performances.

1.4.1 Distance générationnelle et distance générationnelle inversée

La distance générationnelle (GD) mesure la convergence d'un front de Pareto approché S par rapport au front de Pareto optimal PF^*. Pour chaque solution $s_i \in S$, notons d_i la distance Euclidienne entre s_i et le plus proche individu sur PF^*. Le GD peut être définie par :

$$GD(S) = \frac{\sqrt{\sum_{i=1}^{|S|} d_i}}{|S|} \tag{1.14}$$

Il convient de noter que les valeurs objectifs doivent être mises à l'échelle avant du calcul des distances Euclidiennes.

La distance générationnelle inversée (IGD) est une variante du GD. Contrairement au GD, l'IGD considère de la distance depuis le front de Pareto optimal vers le front de Pareto approché. Cette différence est illustrée dans la Figure 1.8. Le GD calcule pour chaque individu sur le front approché, alors que l'IGD s'appuie sur les solutions non-dominées.

Pour chaque solution $s'_i \in PF^*$, notons d'_i la distance Euclidienne entre s'_i et le plus proche individu sur S. L'IGD peut être définie par :

$$IGD(S) = \frac{\sqrt{\sum_{i=1}^{|PF^*|} d'_i}}{|PF^*|} \tag{1.15}$$

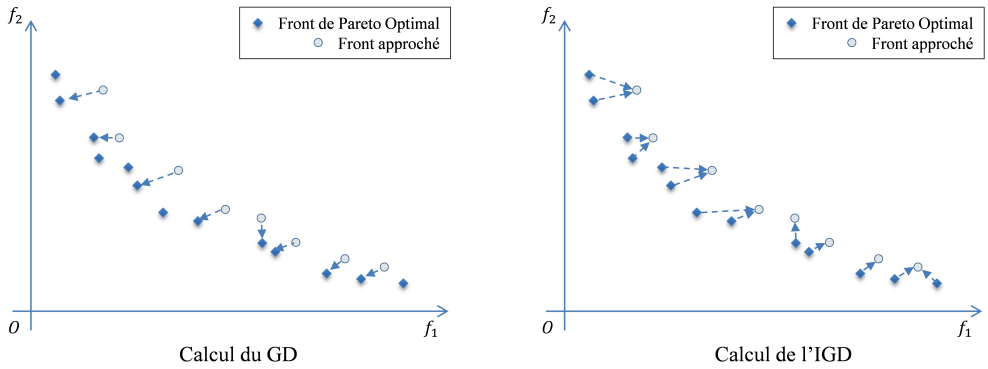

Figure 1.8 – Eléments de calcul pour GD et IGD

Ainsi, elle reflète le niveau de proximité entre S et l'optimum.

Chacune des deux métriques permet de calculer la distance entre les solutions exactes et les solutions approchées. Par conséquent, une valeur proche de 0 implique une bonne qualité selon les deux métriques. Cependant, elles mesurent la distance de façon inversée. En ce qui concerne l'IGD, on part du front de Pareto *optimal* et on considère dans le sens $PF^* \rightarrow S$. Lorsque l'on a IGD = 0, la distance entre chaque solution non-dominée et le front approché est nulle. Ainsi, les deux fronts (exact et approché) sont confondus. En revanche, pour la GD, nous calculons la distance dans le sens $S \rightarrow PF^*$. La mesure GD = 0 implique dans ce cas, que chaque solution dans S est sur le front de Pareto optimal. Toutefois, il n'y a aucune garantie que toutes les solutions non-dominées soient trouvées. Alors, nous ne pouvons pas déterminer si la qualité du front approché est optimisée. Considérons un cas extrême, où PF^* contient 100 solutions et S contient une seule solution non-dominée. On obtiendra GD = 0, tandis que S est loin d'être le front optimal. Au contraire, lorsque l'on considère l'IGD, nous obtiendrons une image plus claire sur la qualité de S.

1.4.2 Métrique de couverture

La métrique de couverture, aussi dit "la métrique C", permet des comparaisons par paire : sa valeur renseigne sur la puissance d'un front non-dominé par rapport à un autre. Soient S_1 et S_2 deux fronts de Pareto, la métrique C calcule pour S_1 (S_2), la proportion de solutions dominées par S_2 (S_1, respectivement) :

$$C(S_1, S_2) = \frac{|s_1 \in S_1 : \exists s_2 \in S_2, s_2 \prec_P s_1|}{|S_1|} \tag{1.16}$$

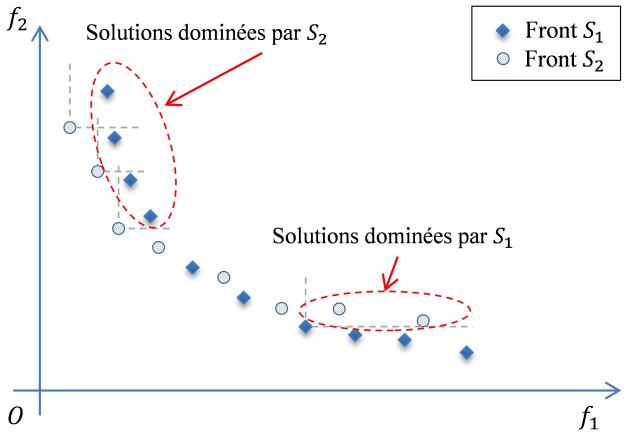

Figure 1.9 – Exemple avec $C(S_1, S_2) = 0.4$ et $C(S_2, S_1) = 0.25$

dont la valeur est comprise dans $[0, 1]$. En outre, nous pouvons calculer le ratio de métriques C afin d'expliciter la comparaison :

$$C_r(S_1, S_2) = \frac{C(S_1, S_2)}{C(S_2, S_1)} \tag{1.17}$$

Par conséquent, $C_r(S_1, S_2) < 1$ signifie que S_1 contient plus de solutions non-dominées par rapport à S_2, aussi plus performant.

Dans la Figure 1.9, nous donnons un exemple de calcul pour la métrique C. Comme on peut le voir dans la figure, 4 parmi 10 solutions dans S_1 sont dominées par au moins une solution de S_2 alors $C(S_1, S_2) = 0.4$. De même, on obtient $C(S_2, S_1) = 0.25$. Par conséquence, nous avons $C_r(S_1, S_2) = 0.4/0.25 = 1.6$, ce qui signifie que S_2 est meilleur que S_1 selon la métrique C.

1.4.3 Hypervolume

L'hypervolume est une des métriques les plus référencées dans les études des problèmes multi-objectif, elle consiste à quantifier l'espace de solution dominée par un front de Pareto. Dans la Figure 1.10, nous montrons l'hypervolume du front donné dans la section 1.3.2.

Comme on peut le voir dans la Figure 1.10, il est nécessaire de borner l'espace de solutions pour calculer l'hypervolume. Pour ce faire, les points Ideal et Nadir sont considérés comme les points de référence théoriques. Le point Ideal (respectivement Nadir), représentant les meilleures valeurs (la plus mauvaises, respectivement) sur chaque objectif, est identifié par toutes les solutions impliquées. Par exemple, lorsque nous comparons deux fronts de Pareto, il est impératif de prendre les mêmes points de référence, comme illustré dans la Figure 1.10.

Le hypervolume permet d'évaluer à la fois la convergence et la divergence des fronts non-dominées. Ceci fait qu'il est une des métriques les plus référencées. Notons que cette métrique

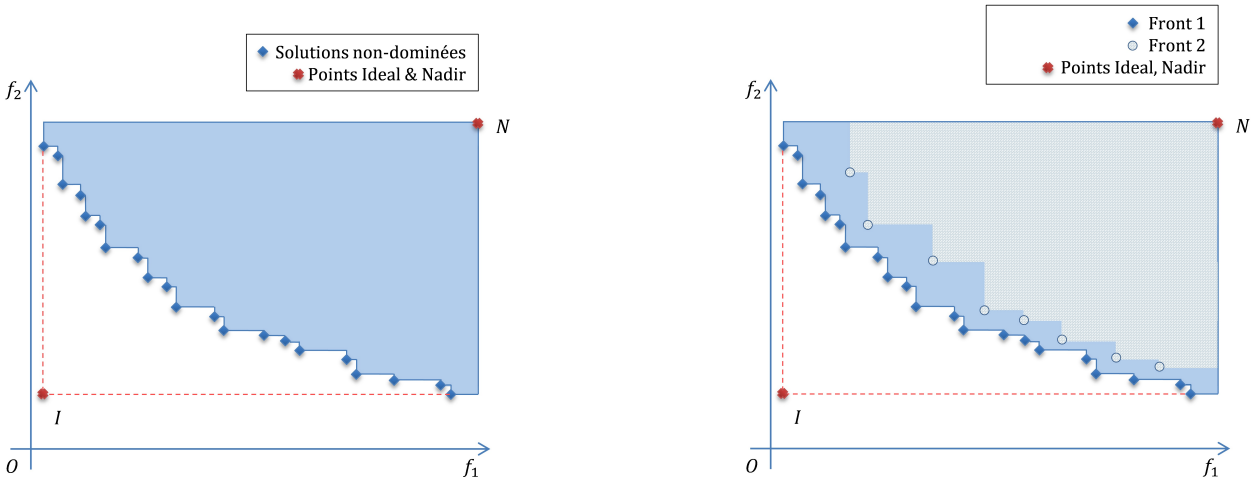

Figure 1.10 – Exemple de la métrique Hypervolume – un front (gauche) et deux fronts (droite)

peut être exprimée par un pourcentage $hv \in [0\%, 100\%]$ – la proportion d'espace dominée par le front étudié. Quand le point Ideal est inclus dans le front, la valeur de hv atteint 100%. Plus généralement, une valeur élevée de hv signifie meilleure qualité du front ; par contre, si le hv est bas, le front correspondant est moins intéressant. Dans l'exemple donné (Figure 1.10), le front 1 est plus performant que le front 2.

1.5 Conclusion

Dans ce chapitre, un état de l'art spécifique sur la littérature du RCPSP, voir celle du MO-RCPSP est proposé. Après avoir introduit le RCPSP, les variantes de ce problème sont classifiés selon différentes caractéristiques. Comme montré dans ce chapitre, le RCPSP est bien documenté en général, alors que le cas multi-objectif MO-RCPSP n'est pas encore beaucoup étudié.

Plus concrètement, en ce qui concerne le MO-RCPSP, les premières études se sont appuyées sur l'application industrielle, avec plusieurs objectifs. Par contre, les études plus récentes se sont concentrées, dans la plupart des cas, sur les problème bi-objectifs. De plus, tout au long de la littérature, peu de travaux ont été consacrés aux fronts de Pareto optimaux. Nous avons aussi remarqué que les travaux portant leur attention le plus souvent sur les tâches ; seulement quelques études ont souligné l'utilisation des ressources, bien que les ressources soient la spécificité du RCPSP.

Pour ces raisons, nous avons proposé dans cette thèse, de résoudre un problème MO-RCPSP (d'abord avec deux critères, élargi ensuite à trois objectifs) avec les méthodes exactes et approchées, pour enrichir la littérature du RCPSP avec un cas proche de l'industrie.

En conséquence, la deuxième partie de ce chapitre est dédiée aux méthodes de résolution et

d'évaluation des problèmes multi-objectif. Sans aller complètement dans les détailles techniques, nous montrerons les contours et certaines caractéristiques de ces méthodes et critères. La plupart des éléments présentés dans cette partie seront détaillés dans les prochains chapitres.

Chapitre 2

Un problème bi-objectif : solutions exactes et approchées

Dans ce chapitre, après la description du problème (section 2.1), nous présentons d'abord les formulations mathématiques du problème bi-objectif (section 2.2). Ensuite, le problème est résolu de manière optimale avec deux méthodes exactes – la méthode à deux phase (section 2.3.1) et la méthode de partitionnement parallèle (section 2.3.2). Dû au fait que le RCPSP soit un problème NP-difficile au sens fort, les méthodes exactes ne permettent que de résoudre des problèmes de petites tailles. Ceci est également montré à travers des expériences numériques (section 2.5.2). C'est pourquoi nous avons implémenté le NSGAII (section 2.4) comme méthode approchée afin de réduire le temps de calcul envisagé.

2.1 Description du problème

Nous considérons dans un premier temps un problème bi-objectif avec la minimisation du makespan et du retard total des tâches. Le makespan est le critère le plus étudié dans la littérature du RCPSP. Pour un problème d'ordonnancement du projet, la date d'achèvement représente directement la productivité. Dans un second temps, nous prenons en compte le retard total des tâches, qui est aussi beaucoup référencé. Il est aussi très souvent considéré dans les entreprises, comme un critère de qualité de projet. Quand nous aboutissons à un programme avec des tâches finissant à l'heure, la satisfaction des clients est assurée. Dans le cas contraire, il y a des insatisfactions avec éventuellement des pénalités. Dans la suite du manuscrit, nous notons le makespan et le retard total respectivement par les symboles f_1 et f_2.

2.2 Modélisation mathématique

Le premier modèle mathématique est proposé par Pristker et al. [56] en 1969. Depuis lors, plusieurs travaux sont consacrés à modéliser le RCPSP. Alvarez-Valdes et Tamarit [52] en 1993, Mingozzi et al. [45] en 1997, Klein [34] en 2000, Artigues et al. [3] en 2003, Carlier et Néron [14] [15] en 2003 et 2007, Sabzehparvar et Seyed-Hosseini [59] en 2008, Bianco et Camaria [8] en 2011, ... La programmation mathématique du RCPSP a toujours été un domaine qui attire de nombreux chercheurs.

Deux modèles mathématiques représentatifs ont été adaptés au problème bi-objectif – le modèle de Pritsker et al. (1969) [56] et celui de Klein (2000) [34]. Certaines notations et leurs significations sont résumées dans le Tableau 2.1.

Tableau 2.1 – Notations

J'	Ensemble de tâches $\{0, ..., n+1\}$
A	Ensemble de contraintes de précédence
R	Ensemble de ressource renouvelable
T	Horizon du projet, une borne supérieure immédiate de $C_{\max}$
p_i	Durée d'opératoire d'une tâche $i \in J'$
d_i	Date de fin souhaitée d'une tâche $i \in J'$
ES_i/LS_i	Date de début au plus tôt / tard d'un tâche $i \in J'$
EF_i/LF_i	Date de fin au plus tôt / tard d'un tâche $i \in J'$
q_r	Capacité périodique d'une ressource $r \in R$
q_{ir}	Demande d'une ressource $r \in R$ par la tâche $i \in J'$
C_i	Date de fin d'une tâche $i \in J'$
C_{max}	Makespan, aussi noté par f_1
T_i	Retard d'une tâche $i \in J'$
M	Un grand nombre

2.2.1 Modèle mathématique 1

Une formulation intuitive et directe du RCPSP est proposée dans les travaux de Pritsker et al. [56], elle permet de bien visualiser les contraintes du problème.

Nous nous inspirons de ce modèle pour modéliser notre problème. La variable de décision clé x_{it} est une variable binaire qui prend la valeur 1 si une tâche $i \in J'$ se termine à un instant t. Ainsi, la date de fin de la tâche i peut être écrite sous la forme $C_i = \sum_{t=0}^{T} t x_{it}$. Pour chaque

tâche $i \in J'$, nous pouvons identifier sa date de fin au plus tôt (notée par EF_i) et sa date de fin au plus tard (notée par LF_i) selon les relations de précédence et les durées de tâches. Par conséquent, le calcul est simplifié sous la forme $C_i = \sum_{t=EF_{n+1}}^{LF_{n+1}} tx_{it}$. De ce fait, le makespan du projet trouvé est $f_1 = C_{n+1} = \sum_{t=EF_i}^{LF_i} tx_{(n+1)t}$ et la somme des retards des tâches est calculée par $f_2 = \sum_{i=0}^{n+1} \max(\sum_{t=EF_i}^{LF_i} tx_{it} - d_i, 0)$. A partir de ces principes, notre problème peut être formulé comme suit :

Fonctions objectives :

$$minimiser \quad f_1 = \sum_{t=EF_{n+1}}^{LF_{n+1}} tx_{(n+1)t} \tag{2.1}$$

$$minimiser \quad f_2 = \sum_{i=0}^{n+1} \max\left(\sum_{t=EF_i}^{LF_i} tx_{it} - d_i, 0\right) \tag{2.2}$$

Sous contraintes :

$$x_{00} = 1 \tag{2.3}$$

$$\sum_{t=EF_i}^{LF_i} x_{it} = 1 \qquad \forall i \in J' \tag{2.4}$$

$$x_{it} = 0 \qquad \forall i \in J', \quad \forall t \notin [EF_i, LF_i] \tag{2.5}$$

$$\sum_{t=EF_i}^{LF_i} tx_{it} \leq \sum_{t=EF_{i'}}^{LF_{i'}} tx_{i't} - p_{i'} \qquad \forall (i, i') \in A \tag{2.6}$$

$$\sum_{i=1}^{n} q_{ir} \sum_{\tau=\max(EF_i - p_i, t-p_i)}^{\min(LF_i - p_i, t)} x_{i\tau} \leq q_r \qquad \forall t \in [0, T], \forall r \in R \tag{2.7}$$

$$x_{it} \in \{0, 1\} \qquad \forall i \in J', \forall t \in [0, T] \tag{2.8}$$

Les fonctions objectives sont données par les équations (2.1) et (2.2) pour le makespan et la somme des retards, respectivement. Le début du projet est indiqué dans l'équation (2.3), la tâche fictive 0 se termine à l'instant 0. L'équation (2.4) garantit que chaque tâche s'exécute exactement une fois dans la fenêtre du temps $[EF_i, LF_i]$. Au contraire, en dehors de cette intervalle, la tâche i ne peut pas être traitée, comme exigé par l'équation (2.5). Les relations de précédence sont exprimées par la contrainte (2.6) – si la tâche i précède la tâche i', alors i' ne peut commencer avant la fin de i. La contrainte (2.7) est la contrainte de ressource, où pour chaque instant t, on compte la somme des ressources des tâches en cours et exige que la consommation ne dépasse pas la capacité. Pour déterminer si une tâche i est en cours à l'instant t, il faut vérifier si t est dans la fenêtre $[C_i - p_i, C_i]$. Pour ce faire, une vérification équivalente est mise en place – pour chaque instant t, on remonte le temps et vérifie si l'instant τ (tel que $x_{i\tau} = 1$) est dans l'intervalle

$[t - p_i, t]$. Pour une autre fois, on peut réduire l'intervalle d'intérêt en tenant compte de EF_i et LF_i. Il est inutile de vérifier pour les instants avant $EF_i - p_i$ et ceux après $LF_i - p_i$, d'où vient la formule donnée dans l'équation (2.7). Et finalement, les variables sont définies comme proposé dans la contrainte (2.8).

2.2.2 Modèle mathématique 2

Le deuxième modèle choisi est celui de Klein (2000) [34]. Au lieu d'exprimer les dates de début des tâches, ce modèle décrit les tâches avec les états. On définit pour chaque tâche $i \in J'$ durant chaque instant $t \in [0, T]$ une variable binaire s_{it}, qui prend la valeur 1 si la tâche i a déjà commencé avant l'instant t, et 0 sinon. Autrement dit, si i se trouve dans l'état "déjà commencé" à l'instant t, alors $s_{it} = 1$. De même, une variable pour décrire l'état de complétion des tâches f_{it} est utilisée : si la tâche $i \in J'$ est déjà terminé avant l'instant $t \in [0, T]$, alors $f_{it} = 1$, sinon $f_{it} = 0$. Par conséquent, le problème bi-objectif peut être modélisé comme le suivant :
Fonctions objectives :

$$minininiser \quad f_1 = T - \sum_{t=0}^{T} f_{(n+1)t} + 1 \tag{2.9}$$

$$minimiser \quad f_2 = \sum_{i=0}^{n+1} \max\left(T - \sum_{t=0}^{T} f_{it} + 1 - d_i, 0\right) \tag{2.10}$$

Sous contraintes :

$$s_{it} \leq s_{i(t+1)} \qquad \forall i \in J', \forall t \in [0, T] \tag{2.11}$$

$$f_{it} \leq f_{i(t+1)} \qquad \forall i \in J', \forall t \in [0, T] \tag{2.12}$$

$$s_{it} = 0 \qquad \forall i \in J', \forall t \in [0, EF_i - p_i - 1] \tag{2.13}$$

$$f_{it} = 0 \qquad \forall i \in J', \forall t \in [0, EF_i - 1] \tag{2.14}$$

$$s_{it} = 1 \qquad \forall i \in J', \forall t \in [LF_i - p_i, T] \tag{2.15}$$

$$f_{it} = 1 \qquad \forall i \in J', \forall t \in [LF_i, T] \tag{2.16}$$

$$\sum_{t=EF_i-p_i}^{LF_i} (s_{it} - f_{it}) = p_i \qquad \forall i \in J' \tag{2.17}$$

$$s_{kt} \leq f_{it} \qquad \forall (i, k) \in A, \forall t \in [ES_i, T] \tag{2.18}$$

$$\sum_{i \in J} q_{ir}(s_{it} - f_{it}) \leq q_r \qquad \forall r \in R, \forall t \in [0, T] \tag{2.19}$$

$$s_{it}, f_{it} \in \{0, 1\} \qquad \forall i \in J', \forall t \in [0, T] \tag{2.20}$$

Les équations (2.9) et (2.10) sont consacrées au makespan et à la somme des retards, respectivement. Pour ce faire, une relation est créée entre la durée de l'horizon et le nombre de périodes après

la fin de i. Étant donné le début de l'horizon à $t = 0$, alors la valeur maximum de $\sum_{t=0}^{T} f_{it}$ est $T+1$. Par exemple, si une tâche i finit à $t = 2$, on a $f_{i0} = 0$, $f_{i1} = 0$ et $f_{it} = 1, \forall\, 2 \leq t \leq T$. Alors nous avons $C_i = (T+1) - \sum_{i=2}^{T} f_{it} = (T+1) - \sum_{i=2}^{T} 1 = (T+1) - (T+1-2) = 2$, ce qui revient à l'équation (2.9). Les contraintes (2.11) – (2.16) décrivent quelques propriétés des variables. Les variables s_{it} et f_{it} sont des fonctions non-décroissantes au cours du temps, comme montré dans les contraintes (2.11) et (2.12). Les contraintes (2.13) – (2.16) permettent de réduire les fenêtres de temps d'intérêt à l'aide de LF et EF que l'on peut calculer en avance. Pour une tâche donnée i, il est sûr qu'elle ne peut pas finir (commencer) avant EF_i ($EF_i - p_i$, respectivement) ; et qu'elle doit déjà avoir fini (commencé) après LF_i ($LF_i - p_i$, respectivement). L'équation (2.17) exige que le nombre périodes où une tâche est dans l'état "déjà commencé mais pas encore fini (donc en cours)" correspond à sa durée. Ensemble avec les contraintes (2.11) et (2.12), elles assurent que les tâches ne sont pas interruptibles. Dans la contrainte (2.18), la relation de précédence est mise en évidence – si la tâche i précède la tâche k, alors k peut seulement commencer quand i est dans l'état fini. La contrainte ressource est formulée dans l'inéquation (2.19). Une tâche i est considérée en cours à l'instant t si $f_{it} - s_{it} = 1$, il faut donc garantir que la consommation totale des tâches en cours ne dépasse pas la capacité. Et finalement, la contrainte (2.20) définit les variables.

Le modèle de Klein (2000) [34] est connu comme étant l'un des modèles les plus performants. Lors de la résolution exacte de notre problème bi-objectif, il est impératif de résoudre une série de problème mono-objectif. Nous allons donc intégrer le modèle présenté dans cette section dans la programmation mathématique pour avoir une meilleure performance.

2.3 Méthodes exactes pour le problème bi-objectif

2.3.1 Méthode à deux phases – TPM

L'idée principale de la TPM est de résoudre une série de problème avec des objectifs pondérés, dont les solutions optimales sont des solutions Pareto optimales du problème bi-objectif. La méthode se déroule en deux temps (d'où vient "Two Phase") – pendant la phase I, les solutions efficaces et supportées (définition voir section 1) sont trouvées ; alors que les solutions efficaces non-supportées ne sont trouvées que dans la phase II.

2.3.1.1 Première phase

La phase I permet non seulement de trouver toutes les solutions supportées mais aussi de réduire l'espace de solution.

Dans un problème bi-objectif, les points Ideal et Nadir peuvent être trouvés en optimisant chacun des objectifs. Comme le montre la figure 2.1, s_1 et s_2 sont obtenues pour la première fois, et elles bornent l'espace de solution dans une zone rectangulaire. Notons le point Ideal par $I = (f_1(s_1), f_2(s_2))$. Afin de trouver les solutions supportées, les sous-problèmes sont créés par les pondérations sur les objectifs f_1 et f_2. Notons f_λ l'objectif pondéré, s_a, s_b ($f_1(s_a) < f_1(s_b)$) les deux solutions choisies pour créer le sous-problème, et P_{ab} le sous-problème. P_{ab} consiste donc à optimiser $f_\lambda = \lambda_1 f_1 + \lambda_2 f_2$, où :

$$\begin{cases} \lambda_1 = f_2(s_a) - f_2(s_b) \\ \lambda_2 = f_1(s_b) - f_1(s_a) \end{cases} \tag{2.21}$$

Sur la figure 2.1, après avoir trouvé s_1 et s_2, elles sont directement choisies pour créer le premier sous-problème :

$$P_{12} : \quad minimiser \;\; f_\lambda = \lambda_1 f_1 + \lambda_2 f_2 \;\; \begin{cases} \lambda_1 = f_2(s_1) - f_2(s_2) \\ \lambda_2 = f_1(s_2) - f_1(s_1) \end{cases} \tag{2.22}$$

Avec cette configuration, s_1 et s_2 font partie des solutions qui minimisent f_λ pour P_{12}. La solution que l'on cherche, différente de s_1 et s_2, se situe dans la zone triangulaire $\triangle s_1 I s_2$ si elle existe (voir Figure 2.1). Pour ces raisons, trois contraintes supplémentaires sont prises en compte :

$$f_1 \neq f_1(s_1) \tag{2.23}$$

$$f_2 \neq f_2(s_2) \tag{2.24}$$

$$f_\lambda \leq f_\lambda(s_1) \tag{2.25}$$

Les contraintes (2.23) et (2.24) garantissent que la nouvelle solution, si elle existe, n'a pas été déjà trouvée auparavant. La contrainte (2.25) met une borne supérieure pour f_λ (trouvée avec s_1 ou s_2), pour que la nouvelle solution soit sûrement supportée. En revanche, si aucune solution n'existe pour ce problème pondéré, nous pouvons conclure qu'il n'y a plus de solution supportée dans $\triangle s_1 I s_2$ mise à part celle déjà connue.

Supposons que s_3 soit la solution trouvée en optimisant P_{12}, deux nouveaux problèmes P_{13} et P_{23} sont donc établis, ainsi de suite. Dans l'exemple présenté sur la figure 2.1, s_3, s_4 et s_5 sont obtenues pour les problèmes P_{12}, P_{13} et P_{32}, respectivement. Il faut noter que les valeurs de λ_1 et λ_2 varient d'un sous problème à un autre. Par exemple, pour P_{43}, nous avons $\lambda_1 = f_2(s_4) - f_2(s_3)$ et $\lambda_2 = f_1(s_3) - f_1(s_4)$. De cette manière, on construit et ensuite on résout les sous-problèmes jusqu'au moment où aucune nouvelle solution n'existe.

A la fin de la phase I, toutes les solutions supportées sont obtenues. Pour l'exemple de la figure 2.1, l'espace d'exploration est réduite à la zone grisée sur la figure 2.2, où se situe les solutions non-supportées.

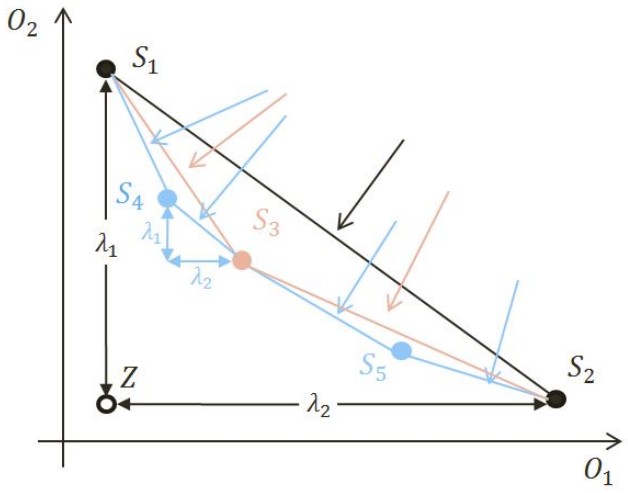

Figure 2.1 – Phase I – méthode d'exploration

2.3.1.2 Deuxième phase

L'espace d'exploration de la phase se situe dans les zones de triangles bornées par chaque couple de solutions supportées adjacentes (Figure 2.2). La méthode générale utilisée pendant cette phase est similaire à celle présentée dans la phase I. Cependant, quelques ajustements sont mis en place pour chercher les solutions non-supportées. A la fin de la phase I, on sait que pour une zone

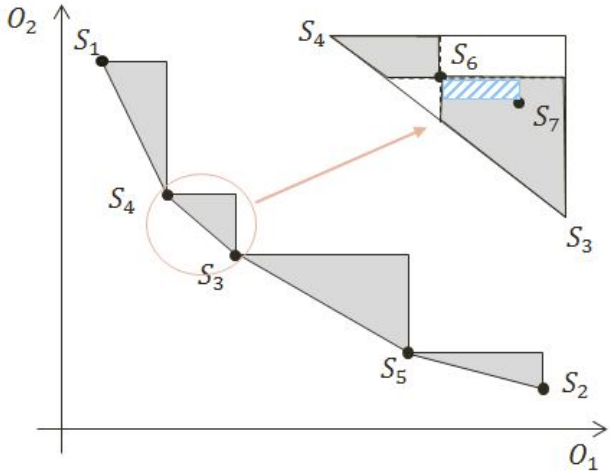

Figure 2.2 – Phase II – méthode d'exploration

triangulaire encadrée par deux solutions adjacentes, aucune autre solution ne permet de minimiser le sous-problème correspondant. Alors pour la phase II, la contrainte (2.25) n'est plus valide. De plus, au lieu d'éliminer certains points précis comme dans les contraintes (2.23) et (2.25), les bornes supérieures sont considérées. Par exemple, dans la partie zoomée et bornée par s_3 et s_4,

nous avons :

$$f_1 < f_1(s_3) \tag{2.26}$$

$$f_2 < f_2(s_4) \tag{2.27}$$

Supposons que s_6 soit la première solution trouvée dans ce triangle, l'espace de recherche partielle peut être réduit selon la dominance de Pareto – la zone A en tant que région non-dominée en commun de s_4 et s_6 ; la zone B est celle de s_5 et s_6. Par conséquent, la zone d'intérêt est réduite à la région foncée dans le zoom (voir Figure 2.2). De nouveaux sous-problèmes sont créés par les couples (s_4, s_6) et (s_5, s_6) avec les valeurs objectives bornées. Généralement, dés qu'une nouvelle solution est découverte, elle enchaîne de nouveaux sous-problèmes avec potentiellement de nouvelles solutions. Il convient de noter que la région encadrée par deux solutions non-supportées est sous forme de rectangle au lieu de triangle – par exemple, celle par s_6 et s_7. Chaque zone rectangulaire peut être considérée comme une nouveau problème (au lieu d'un sous-problème) avec des contraintes spécifiques sur les valeurs objectives. Il faudra donc procéder à une pseudo-phase I avant de lancer les démarches de phase II.

2.3.1.3 Exploration des solutions

Dans la TPM, nous devons chercher les solutions à l'intérieur de l'espace borné au cours de chacune des phases. En réalité, la première phase peut être considérée comme une exploration dans une région rectangulaire bornée par la solution Ideal et Nadir. Dans la phase II, les zones engendrées sont explorées par les couples de solutions qui peuvent être de forme triangulaire ou rectangulaire. Malgré leur différence, le même schéma d'exploration a été adopté pour toutes les zones à explorer.

En se référant au fait précédant, quand on trouve une nouvelle solution, deux sous-problèmes sont immédiatement établis. Notons s_c la nouvelle solution obtenue par P_{ab}, les deux sous-problèmes sont alors P_{ac} et P_{cb}. Les solutions s_a et s_b sont des "solutions parents" de s_c. Notons S l'ensemble de solutions triées dans l'ordre croissant de f_1, s_c est insérée par conséquent entre s_a et s_b. Notons $p(s)$ la position de la solution s dans l'ensemble S, on peut en déduire que $p(s_c) = p(s_a) + 1$. En même temps, $p(s_b)$ est décalée par un, comme toutes les autres solutions derrière s_b. En définitif, trois positions dynamiques peuvent être associées pour chaque solution – une pour elle-même et deux pour ses solutions "parents". Ceci permet de marquer l'état d'exploration des solutions. Une solution est dite "complètement explorée" si tous les deux sous-problèmes constitués par ses "parents" et sont résolus.

Notons S_E l'ensemble des solutions complètement exploré et initialisé par $\varnothing$. Pour choisir une solution à étudier (à partir de laquelle on crée des sous-problèmes et les résout), nous considérons

la première solution rencontrée s, telle que $s \in S \cap s \notin S_E$. Une fois s choisie, on doit s'assurer que les deux sous-problèmes associés à s soient résolus avant de passer à une autre solution. Après avoir considéré les deux sous-problèmes, s est ajoutée à l'ensemble S_E. Mais ceci ne signifie pas forcément que tous les sous-problèmes en relation avec s sont étudiés. Il est possible que s soit réconvoquée par les sous-problèmes créés par des nouvelles solutions. Les deux solutions extrêmes s_1 et s_2 sont en effet un cas spécial, car elles n'ont pas de solutions "parents". Elles sont donc ajoutées directement dans S_E après avoir considéré ce sous-problème qui "démarre" la suite de la méthode.

La résolution du problème est accomplie lorsque S_E contient tous les membres de S. Autrement dit, toutes les solutions trouvées sont complètement étudiées, et on ne trouve aucune autre solution. Pour l'exemple utilisé dans cette section (voir Figure 2.2), l'ordre d'exploration de solutions est :

$$\begin{cases} \text{Phase I} : P_{12}(s_3), P_{13}(s_4), P_{32}(s_5), P_{14}(\varnothing), P_{43}(\varnothing), P_{35}(\varnothing), P_{52}(\varnothing) \\ \text{Phase II} : P_{14}(\varnothing), P_{43}(s_6), P_{46}(\varnothing), P_{63}(s_7), P_{67}(\varnothing), P_{73}(\varnothing), P_{35}(\varnothing), P_{52}(\varnothing) \end{cases}$$

où on marque la nouvelle solution issue d'un sous-problème entre les parenthèses.

A la fin de la phase II, toutes les solutions non-supportées sont trouvées. Avec celles obtenues par la phase I, nous avons le front de Pareto optimal.

2.3.2 Méthode de partitionnement parallèle – PPM

Dans la section 1, la PPM, comme le TPM, résout une série de problèmes mono-objectifs. Elle se déroule en deux temps. Pour différencier avec la TPM, on dit pour la PPM "étape 1" et "étape 2". Lors de l'étape 1, l'espace de solution est découpé en plusieurs parties selon un objectif (f_1 par exemple). Ce qui crée plusieurs intervalles de contraintes et transforme cet même objectif en contrainte. Par conséquent, nous n'avons qu'à optimiser l'autre critère (f_2 dans ce cas). L'étape 2 de la PPM explore ainsi l'espace de solutions réduite par les solutions de l'étape 1. Les mêmes démarches d'exploration utilisées dans la TPM peuvent être appliquées.

2.3.2.1 Première étape

Pour démarrer la méthode, l'étape commence par résoudre les deux problèmes mono-objectif et chercher les deux solutions extrêmes s_1 et s_2. Ainsi, l'espace de recherche est encadrée, dans le cas de la TPM. La PPM, elle, propose de découper l'espace de solution en N_{PPM} tranches de même taille et de faire la recherche dans ces sous-espaces.

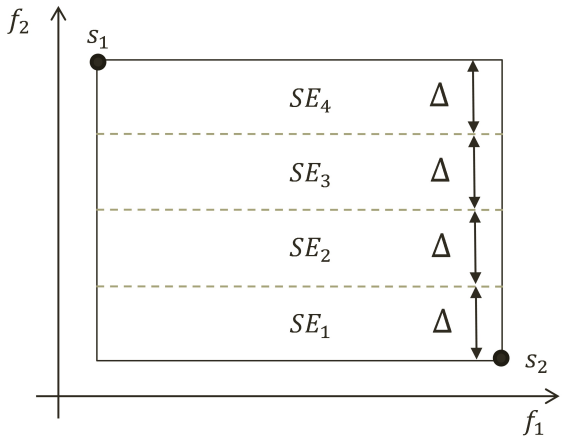

Figure 2.3 – Découpe de l'espace de recherche avec $N_{PPM} = 4$

Reprenons le même exemple utilisé dans la section précédente. Après avoir trouver s_1 et s_2, l'espace de recherche est divisé en plusieurs parties avec $N_{PPM} = 4$. Ainsi, on peut identifier plusieurs sous-espaces, notées par $SE_i, i = 1, \ldots N_{PPM}$ (Figure 2.3). Dans l'exemple donnée, l'espace de recherche est découpée selon l'objectif f_2. De ce fait, toutes les SE_i partagent au départ les mêmes bornes sur f_1, telles que

$$f_1(s_1) < f_1 <= f_1(s_2) \tag{2.28}$$

Au contraire, sur f_2, chaque SE_i ($i = 1, \ldots, N_{PPM}$) a ses propres bornes. Par exemple, pour SE_1, nous avons

$$f_2(s_1) - \Delta < f_2 <= f_2(s_1) \tag{2.29}$$

où Δ désigne à l'écart entre les bornes supérieure et intérieure de f_2 pour i quelconque :

$$\Delta = \frac{f_2(s_1) - f_2(s_2)}{N_{PPM}} \tag{2.30}$$

Plus généralement, dans un certain sous-espace SE_i, f_2 est bornée comme suit :

$$LB_i < f_2 <= UB_i \begin{cases} LB_i = f_2(s_2) & \text{si } i = 1 \\ UB_i = LB_i + \Delta & \forall i = 1, \ldots, N_{PPM} \\ LB_i = UB_{i+1} & \forall i = 2, \ldots, N_{PPM} \end{cases} \tag{2.31}$$

Une fois les SE_i fixés, ils seront explorés un par un. Pour un espace ES_i donné, on cherche à minimiser l'objectif f_1 tout en respectant les bornes sur f_2 (figure 2.4). Cela signifie que nous voulons résoudre le problème tel que :

$$\begin{cases} minimiser \quad f_1 + \varepsilon f_2 & \\ LB_i < f_2 <= UB_i & \forall i = 1, \ldots, N_{PPM} \\ \text{sous contraintes (2.11) – (2.20)} & \end{cases} \tag{2.32}$$

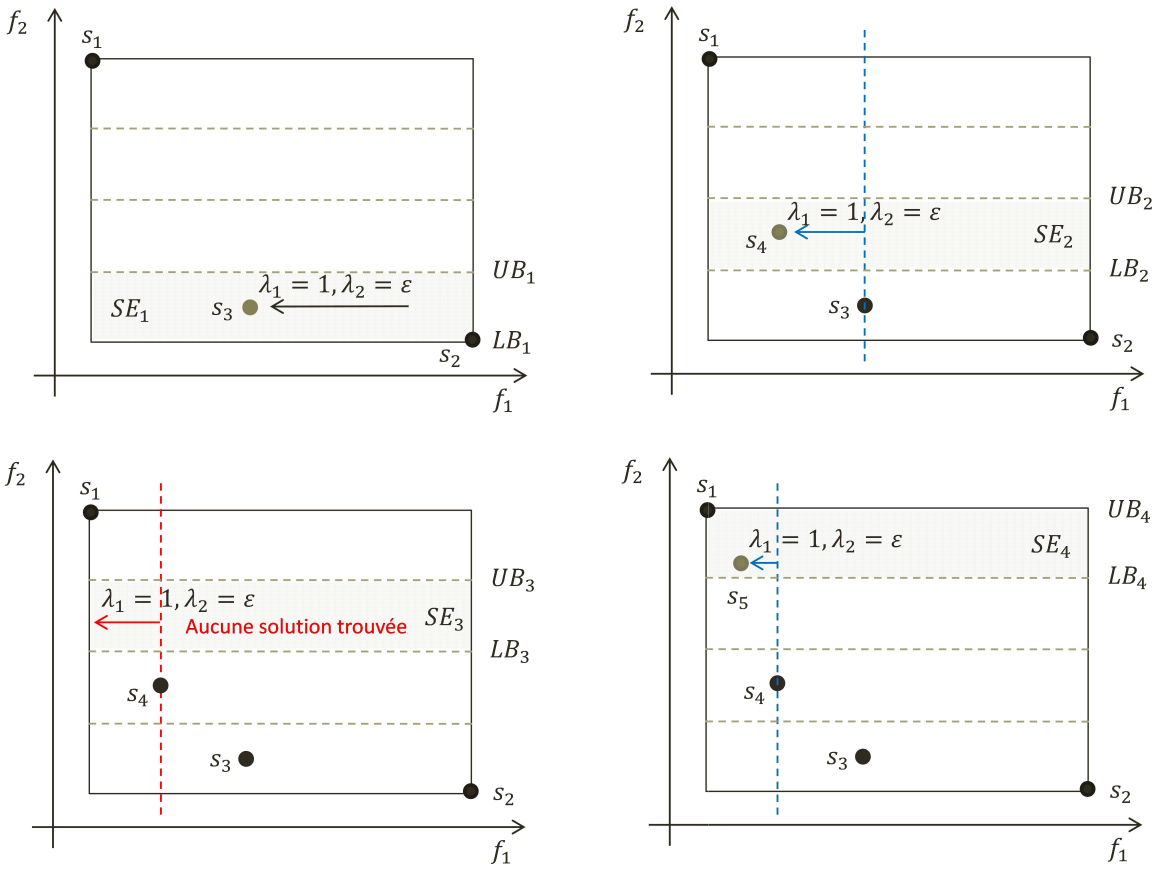

Figure 2.4 – Exploration des sous-espaces

où ε est une petite valeur pour assurer que la solution trouvée n'est pas dominée. En effet, il est possible d'avoir deux solutions s_a et s_b telles que $f_1(s_a) = f_1(s_b)$ et $f_2(s_a) \neq f_2(s_b)$. Dans ce cas, si aucune précision n'était imposée sur f_2, on pourrait trouver s_a tout comme s_b quand on minimise f_1 ; alors que l'on ne peut pas garantir que la solution prise en compte est non-dominée. Toutefois, ε doit être suffisamment petite pour ne pas dépasser ce qu'il faut. Dans nos travaux, la valeur fixée $\varepsilon = \frac{1}{f_2(s_1)+f_2(s_2)+1}$ pour les deux raisons suivantes : l'espace de recherche dans notre problème est l'espace des entiers, il suffit alors d'avoir $\varepsilon f_2 < 1$. Pour ce faire, nous prenons la plus grande valeur possible sur f_2 (donnée par s_1) et en rajoute $f_2(s_2) + 1$ pour être le dénominateur afin d'avoir $\varepsilon f_2 < 1$. Une fois tous les sous-espaces explorées, nous obtenons l'espace de solution réduite par $s_1 - s_5$ (figure 2.5).

2.3.2.2 Deuxième étape

L'étape 2 de la PPM est similaire avec la phase II de la TPM, il s'agit de la recherche de solutions qui se situent dans la zone réduite lors de la première étape. En revanche, comme les solutions non-trouvées sont susceptibles d'être supportées comme non-supportées, les zones d'intérêt restent

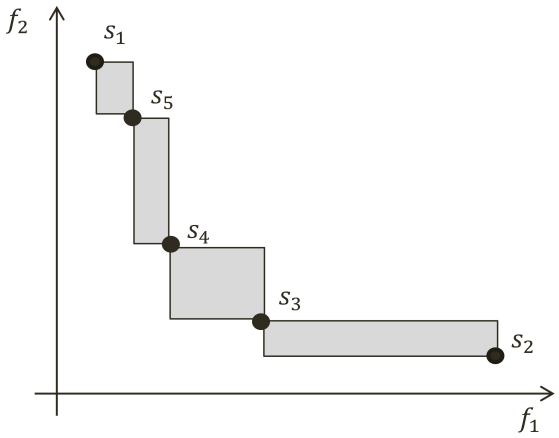

Figure 2.5 – PPM – solutions trouvées et espace réduite de l'étape 1

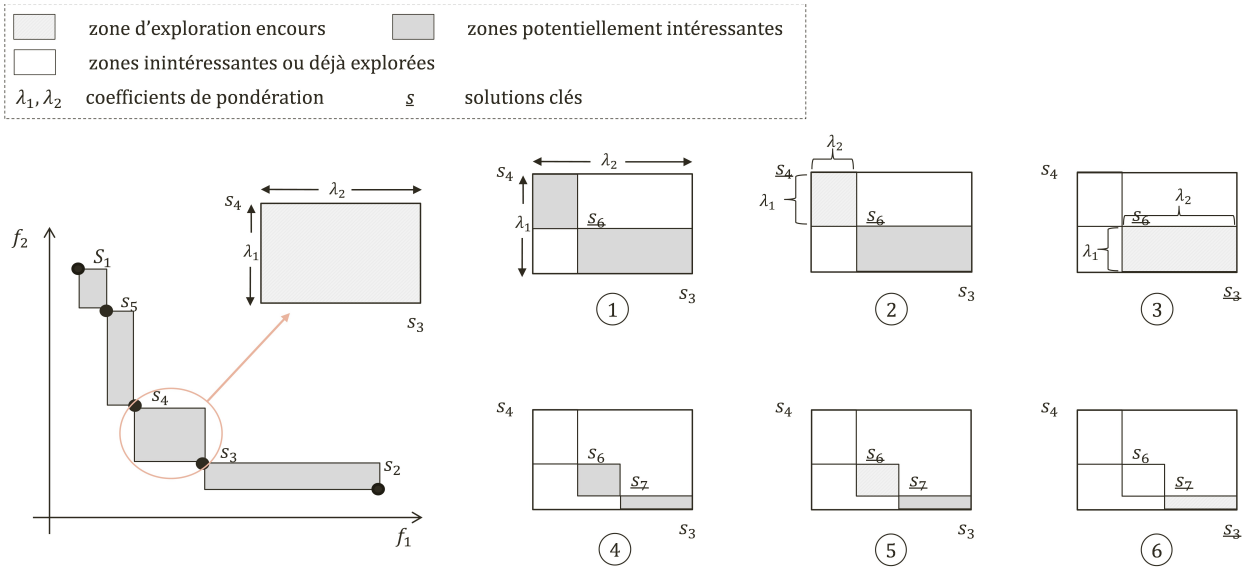

Figure 2.6 – PPM – solutions trouvées et espace réduite de l'étape 1

toujours sous forme de rectangle au lieu de triangle.

Les principes de calculs sont déjà expliqués dans la section 2.3.1. Dans cette section, un exemple est donné. Supposons qu'il existe deux autres solutions dans l'espace encadré par s_3 et s_4, nous présentons les étapes d'exploration de celle-ci sur la figure 2.6. En complément du graphe, la synthèse suivante est donnée :

$$P_{43}(s_6), P_{46}(\varnothing), P_{63}(s_7), P_{67}(\varnothing), P_{73}(\varnothing)$$

Une partie de la PPM partage les mêmes principes section 2.5.2, nous allons comparer leurs performances lors du test.

2.4 Méthode approchée – Non-dominated Sorting Genetic Algorithm II

2.4.1 Principe de la méthode

Le NSGAII suit le schéma général des algorithmes génétiques. Du fait qu'elle est une méthode évolutionnaire à population, elle prend en compte une population de N individus. Chaque individu est identifié par son *chromosome*. L'algorithme commence par une génération des solutions initiales. En outre, des opérateurs de *reproduction* sont utilisés afin de créer de nouveaux individus (aussi appelés "Enfants") à partir des solutions existantes (aussi appelées "Parents"). Pour maintenir la taille de la population, N individus sont choisis parmi l'ensemble des *Parents + Enfants*, à l'aide de la procédure de la *sélection*. Ces solutions choisies constituent ainsi la nouvelle génération, qui sont à nouveau considérées comme des *Parents* pour la reproduction. Reproduction, sélection, ..., cette boucle est répétée tant que la condition d'arrêt n'est pas satisfaite. Dans nos travaux, l'algorithme se termine quand le nombre de génération atteint n_G.

2.4.2 Codage du chromosome et construction de solution

Chaque individu est identifié par son chromosome. Nous adoptons un codage de "liste de priorité", basé sur l'affectation "position - tâche", pour le problème étudié. Notons V un chromosome et X_i le $i^{\text{ième}}$ gène, une liste de taille $n+2$ (dont n tâches non-fictives et 2 tâches fictives) est créée ; chaque gène représente le numéro de tâche affectée à la position correspondante. Par conséquent, le chromosome peut être formulé, de façon générale, par $V =< 0, X_i, n+1 >$ avec $i \in \{1, ..., n\}$, comme on peut le voir sur la figure 2.7. De plus, la position d'une tâche $i \in J'$ dans le chromosome V peut être notée par $p(i)$. Pour un individu quelconque, $p(0)$ et $p(n+1)$ sont fixes car ces tâches représentent respectivement le début et la fin du projet. La liste de priorité correspond à l'ordre dans lequel on affecte la date de début de chaque tâche. De ce fait, si une tâche $a \in J$ est devant une tâche $b \in J$ dans le chromosome, nous devons décider de la date de début pour a avant de considérer b. Cela implique également que les X_i doivent respecter toutes les contraintes de précédence pour éviter l'infaisabilité.

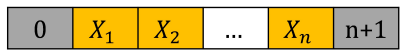

Figure 2.7 – Représentation générale du chromosome

Reprenons l'exemple utilisé dans le Chapitre 1, avec 5 tâches non-fictives. Dans le tableau 2.2, les données sont reprises et ensuite complétées par les dates de fin des tâches souhaitées.

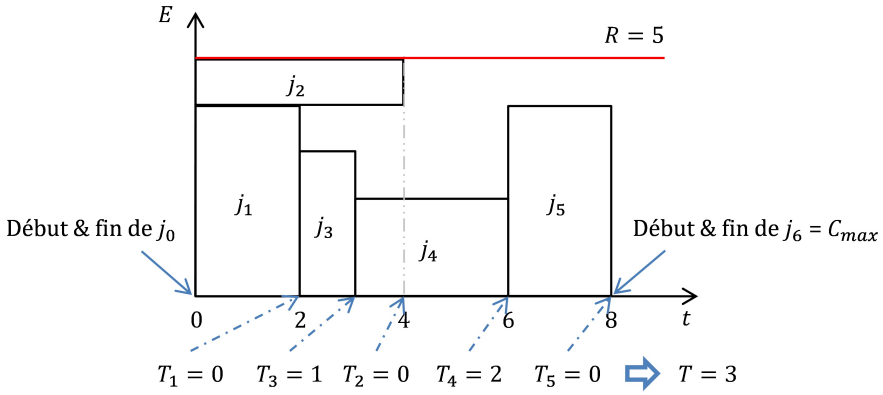

Figure 2.9 – Solution s_e avec $f_1 = 8, f_2 = 3$

Lorsque l'on construit un chromosome, seul les $Pred_i$ sont des contraintes effectives. Nous mettons l'accent sur une solution d'exemple, notée par s_e, que l'on va re-solliciter plusieurs fois dans la suite de ce rapport. La séquence de tâche prise en compte est 0-1-2-3-4-5-6 (la même est utilisée dans le Chapitre 1), dont le chromosome est représenté sur la figure 2.8.

Figure 2.8 – Chromosome de s_e

Tableau 2.2 – Un exemple avec 7 tâches

Tâche (i)	0	1	2	3	4	5	6
$Pred_i$	-	0	0	1	0	2	3,4,5
p_i	-	2	4	1	3	2	-
r_i	-	4	1	3	2	4	-
d_i	-	5	5	2	4	8	-

Pour rappel, lorsque l'on crée une solution à partir de la séquence de tâche, toutes les tâches commencent le plus tôt possible sous conditions. Pour une tâche $i \in J$, il faut tout d'abord vérifier que tous ses prédécesseurs soient finis ; ensuite, on doit assurer qu'il y a suffisamment de ressource pendant chaque période où la tâche i est en cours de traitement. La solution est présentée sur la figure 2.9 : les valeurs des objectifs sont mises en évidence. Cette même séquence est aussi utilisée dans le Chapitre 1 où nous avons déroulé la construction de solution en détail. Prenons ici l'exemple de la tâche 5, qui est considéré en dernier – dans un premier temps, elle doit attendre que son prédécesseur soit fini avant de commencer (la tâche 2) ; d'ailleurs, il lui faut suffisamment de ressource pendant les périodes en cours. Par conséquent, elle ne peut commencer qu'à l'instant 6. Tenant en compte des dates de fin souhaitées des tâches, s_e propose un makespan $f_1 = 8$ et une somme de retard des tâches $f_2 = 3$.

Il convient de noter que pour notre problème, différents chromosomes peuvent conduire à la même solution. Comme plusieurs tâches peuvent être exécutées en même temps, l'ordre attribué par le chromosome aux tâches qui n'ont pas de relation de précédence n'est pas forcément l'ordre d'exécution dans la solution finale. Par exemple, pour le cas décrit ci-dessus, si on change l'ordre

de la tâche 1 et la tâche 2 dans le chromosome, le résultat reste le même. Il n'y a pas de relation de précédence entre les deux tâches car la disponibilité de ressource est suffisante pour soutenir les deux tâches en même temps.

2.4.3 Croisement et mutation

La procédure de reproduction permet de créer de nouveaux individus à partir de la population actuelle. Pour ce faire, les opérateurs du croisement (section 2.4.3.1) et de la mutation (section 2.4.3.2) sont mis en œuvre. Notons P la population "parents" et Q la population "enfants" issue de la reproduction par la suite.

2.4.3.1 Croisement

Avec un couple de parents choisis, le croisement s'effectue avec une probabilité c_r (*crossover rate*). Comme nous l'avions vu précédemment, différents chromosomes peuvent finir par les mêmes valeurs objectives. Nous adoptons donc deux opérateurs de croisement pour le problème – avec un point de croisement (*one-point crossover*) et avec deux points de croisement (*two-point crossover*), afin d'avoir plus de combinaisons possibles des gènes.

Les deux points de croisement sont notés par $CP1$ et $CP2$, où $CP1$ est par défaut devant $CP2$ ($CP1 < CP2$). Les points de croisements sont aléatoirement choisis, tels que $CP1 \in [1, n-1]$ et $CP2 \in [2, n]$. Dans le croisement à un point, $CP2$ est systématiquement choisi à la position $n+1$ (donc non-effectif). Les chromosomes sont désormais divisés en plusieurs parties.

Notons F, M, S, D le Père, la Mère, le Fils et la Fille, respectivement. Sur la figure 2.10, un exemple du croisement à deux points est donné. Pendant le croisement, le Fils (la Fille) reprend les mêmes gènes de ceux du Père (de la Mère, respectivement) avant $CP1$ et après $CP2$; pour les positions entre $CP1$ et $CP2$, le Fils (la Fille) suit l'ordre respectif des tâches comme celui de la Mère (du Père, respectivement). Ce mécanisme permet de garantir également que toutes les contraintes de précédence sont respectées.

2.4.3.2 Mutation

Après le croisement, la mutation est appliquée sur les enfants avec une probabilité m_r (*mutation rate*). Nous utilisons la technique d'*insertion* comme opérateur de mutation. Le Fils et la Fille mutés sont notés respectivement S' et D'.

Une tâche de mutation j^m est d'abord choisie de façon aléatoire. A partir de cela, nous pouvons déterminer son intervalle libre $IL(j^m)$, dans lequel j^m peut glisser vers n'importe quelle position

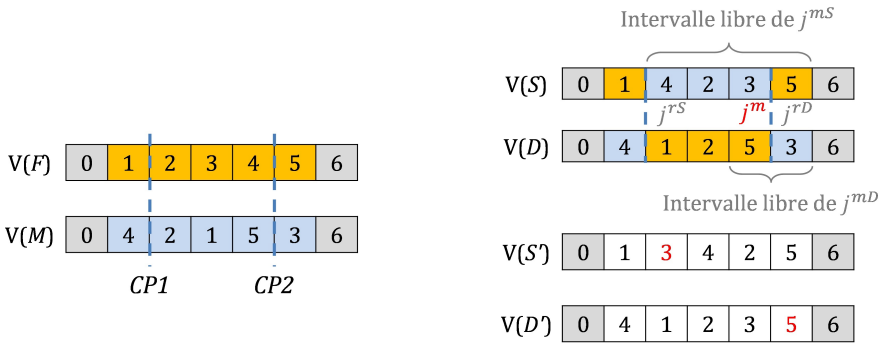

Figure 2.10 – Un exemple de croisement (en haut à droite) et de mutation (en bas à droite)

sans violer aucune contrainte de précédence. Cette intervalle est engendrée par le dernier prédécesseur fini (noté par j^{dp}) et le premier successeur commencé (noté par j^{ps}) de j^m. Autrement dit, j^m peut se déplacer librement dans l'intervalle $IL(j^m) = [p(j^{dp}) + 1, p(j^{ps}) - 1]$. Nous choisissons par la suite une tâche de référence $j^r \neq j^m$ dans $IL(j^m)$. Si j^r est avant (après) j^m dans le chromosome, on déplace j^m et le met juste derrière (devant, respectivement) j^r. Les autres tâches dans $IL(j^m)$ suivront toujours leur ordre respectif.

Reprenons l'exemple déroulé lors du croisement, nous illustrons la mutation sur la même figure 2.10. Nous fixons la position 5 pour avoir la tâche de mutation pour S et D, ce que fait $j^{mS} = 3$ et $j^{mD} = 5$. Nous pouvons en déduire donc $IL(j^{mS}) = [3, 6]$ et $IL(j^{mD}) = [5, 6]$. De ce fait, pour $V(S)$, la position 3 est aléatoirement choisie donc $j^{rS} = 4$. Pour avoir S', nous déplaçons ainsi la tâche 3 juste devant la tâche 4, et le reste des tâches suivent leur ordre respectif dans S. De même, j^{mD} est choisie dans $IL(j^{mD})$. Comme la tâche 3 est la seule tâche répondant aux conditions, elle est naturellement choisie. La tâche 5 est donc mise derrière la tâche 3 dans $V(D')$.

2.4.4 Sélection – "Non-dominated Sorting"

Avec le croisement et la mutation, le processus de reproduction permet de créer les nouveaux individus, ainsi que la population enfants Q. Afin d'avancer vers la génération suivante, une sélection est mise en place. Dans le cadre de Non-dominated Sorting Genetic Algorithms (NSGAs), la sélection se base sur la dominance de Pareto (définition voir la section 1).

L'objectif de cette étape est de choisir N individus parmi l'ensemble des parents et des enfants. Pour ce faire, les fronts non-dominés sont construits. Notons F_1, le premier front non-dominé qui contient alors les meilleurs solutions de la génération. Les solutions sur F_1 sont mutuellement non-dominées ; et toute solution sur F_1 n'est dominée par aucune autre solution de la population. Ensuite, on exclut F_1 de la population et on cherche à nouveau les meilleures solutions : on obtient F_2. Ainsi de suite, plusieurs fronts non-dominés $F_i, i \in \{1, 2, \ldots\}$ sont trouvés et choisis pour

la prochaine génération. Tant que le nombre de solutions choisies n'atteint pas N, la taille de population, nous continuons à explorer les fronts.

Dans la plupart des cas, lorsque le dernier front intéressant F_l est impliqué, toutes les solutions ne sont pas choisies. Dans ce cas, le NSGAII les distingue selon la répartition des solutions dans l'espace objectif et préfère celle qui se situent aux zones moins visitées. La mesure calculée par le NSGAII est appelée *crowding distance*, qui signifie distance d'encombrement. Elle évalue une solution s selon l'espace occupé par son voisin ; cette valeur est notée $d_C(s)$. Dans l'algorithme 1, ci-après, le calcul de *crowding distance* est résumé. Cet algorithme permet de calculer la d_C pour les problèmes bi-objectifs, ainsi que pour les problèmes multi-objectifs. Pour une meilleure compréhension, un exemple bi-objectif est présenté sur la figure 2.11.

Algorithm 1 Choix des solutions sur F_l selon *crowding distance*

Entrées : F – un front non-dominé
n_l – nombre de solutions à choisir
Sortie : F' – ensemble de solutions choisies
Variables : n_F – nombre de solutions sur F
s – une solution sur le front F
$d_C(s)$ – *crowidng distance* de $s \in F$
$s_{[i]}$ – la $i^{\text{ième}}$ solution selon l'ordre trié (à préciser dans l'algorithme)
$f_k^{\min}, f_k^{\max}$ – valeurs minimale et maximale sur l'objectif $k = \{1, 2, \ldots, K\}$

Début
Pour $s \in F$ **faire**
  $d_C(s) = 0$ // Initialisation
Fin pour

Pour $i = 1, \ldots, K$ **faire**
  Trier les solutions sur F selon l'ordre lexicographique de f_k et noter les solutions par $s_{[1]}, s_{[2]}, \ldots, s_{[n_F]}$
  $d_C(s_{[1]}) = d_C(s_{[n_F]}) = \infty$ // solutions extrêmes
  Pour $i = 2, \ldots, n_F - 1$ **faire**
    $d_C(s_{[i]}) = d_C(s_{[i]}) + f_k(s_{[i-1]}) + f_k(s_{[i+1]})$ // solutions non-extrêmes
  Fin pour
Fin pour

Trier les solutions selon l'ordre décroissant de d_C et noter les solutions par $s_{[1]}, s_{[2]}, \ldots, s_{[n_F]}$
$i = 1$
Tant que $|F'| < n_l$ **faire**
  $F' = F' \cup \{s_{[i]}\}$
  $i = i + 1$
Fin tant que

Retourner F'
Fin

Dans l'algorithme 1, le cas général (donc multi-objectif) est traité. Dans un premier temps, les d_C des solutions sont initialisés à 0. Ensuite, cette valeur sera cumulée par rapport à chaque critère. Pour chaque objectif $k \in \{1, \ldots, K\}$, les solutions sont triées selon un ordre croissant (ou décroissant) de leurs valeurs sur f_k. Elles sont ensuite notées $s_{[i]}$ selon l'ordre de tri. Pour les

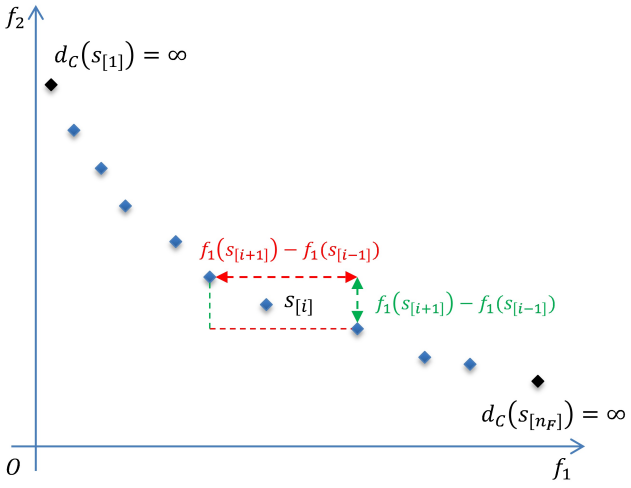

Figure 2.11 – Un exemple de crowding distance

deux solutions extrêmes $s_{[1]}$ et $s_{[n_F]}$, leurs *crowding distances* sont mis à l'infini (alors elles seront certainement choisies). Pour une solution $s_{[i]}$ non-extrême, on calcule la différence entre la solution juste avant et après elle ($s_{[i-1]}$ et $s_{[i+1]}$, respectivement) sur f_k. Cette différence est ensuite rajoutée à $d_C(s_{[i]})$. Après avoir parcouru tous les objectifs, on obtient les d_C finales de chaque individu. Pour choisir parmi elles, les solutions sont triées dans l'ordre décroissant de leurs d_C et celles avec les plus grandes valeurs prioritaires.

Plus particulièrement, dans le cas bi-objectif, pour deux solutions s_a et s_b sur le même front, $f_1(s_a) < f_1(s_b)$ implique directement que $f_2(s_a) > f_2(s_b)$ et verse verso, selon la définition de la dominance de Pareto. Pour cette raison, pendant le tri des solutions, l'ordre croissant sur un objectif revient à l'ordre décroissant sur le deuxième objectif. Le calcul des *crowding distances* nécessite, en réalité, un seul tri pour les deux critères. Comme le montre la figure 2.11, pour une solution $s_{[i]}$, on peut calculer $d_C(s_{[i]})$ selon les deux solutions près d'elle. La valeur obtenue est aussi appelée "cuboïde".

2.5 Tests numériques

2.5.1 Benchmarks et génération des données

Le benchmark le plus référencé dans la littérature du RCPSP est le *KSD set*, aussi appelé le PSPLIB [36], créé par Kolisch et al. en 1997. A partir de différents paramètres, plusieurs groupes d'instances sont générés. La taille des problèmes varie entre 30 tâches et 120 tâches.

Le RCPSP est un problème NP-difficile dans le sens fort, c'est pourquoi les solutions exactes peuvent être trouvées seulement pour les instances de petite taille. Par exemple, dans le RCPSP, la

solution exacte n'est obtenue pour au maximum des instances avec tâches. Comme expliqué dans la section 2.3.1 et la section 2.3.2, pour le problème multi-objectif, il faudra résoudre une série de problèmes mono-objectif. Cela signifie que les solutions inaccessibles en mono-objectif rend la résolution en multi-objectif impossible. De ce fait, nous nous concentrons sur les problèmes de petites tailles dans ce chapitre. Plus précisément, nous commencerons par les problèmes avec 10 tâches et élargissons par la suite.

Les instances du PSPLIB sont caractérisées par une complexité (voir définition en fin de paragraphe) du réseau plutôt élevée : elle est fixée à 1.5, tâches fictives non comprises. Ceci limite fortement le nombre de solutions potentielles, surtout pour les petites instances. Dans les études multi-objectif, le but est d'offrir aux décideurs un ensemble de solutions qui sont Pareto-équivalentes. Une complexité élevée rend le problème potentiellement moins intéressant. D'ailleurs, les dates de fin souhaitées générées dans les instances KSD sont attribuées aux projets au lieux des tâches. Par conséquent, nous avons choisi de créer des nouvelles instances de tests, en s'inspirant des benchmarks dans la littérature et leurs techniques utilisés. Plus concrètement, nous adoptons plusieurs paramètres lors de la génération des données :

- Complexité du réseau (*Net Complexity*, NC) : ratio entre le nombre de tâches et le nombre de relations de précédence
- Taux de demande (*Demand Rate*, DR) : proportion des demandes de ressources q_{ir} avec $i \in J, r \in R$
- Facteur de demande (*Demand Factor*, DF) : mesure pour quantifier les demandes de ressources. Pour une demande q_{ir} valide (supérieure à 0), la valeur est choisie aléatoirement dans l'intervalle $[1, DF * q_r]$ selon la distribution uniforme

De plus, pour générer les dates de fin souhaitées, nous adaptons une notion de "*delay proportion*" utilisé dans les études de Ballestin et al. (2006) [6] :

- Proportion de délai (*Delay Proportion*, DP) : proportion de tâches dont les dates de fin souhaitées sont fixées à leurs dates de fin au plus tôt, calculées selon le graphe de précédence et les durées des tâches

Pour une tâche $i \in J$, si sa date de fin souhaitée d_i n'est pas définie au plus tôt possible, nous choisissons une valeur selon la loi uniforme,

$$d_i \sim EF_i + U\left[1, \frac{\sum_{i \in J} p_i}{n} \times \frac{n_i^s}{n_i^p} \times \frac{\sum_{i \in J} \sum_{r \in R} q_{ir}}{\sum_{r \in R} q_r}\right]$$

où n est le nombre de tâches non-fictives ; n_i^s et n_i^p représentent respectivement le nombre de prédécesseurs et successeurs d'une tâche $i \in J$. En effet, cette équation prend en compte trois facteurs – la durée des tâches en moyennes, le rôle dont la tâche i joue dans le réseau de précédence, et finalement le nombre de ressource demandé en moyenne.

Afin de simuler les différentes configurations, nous considérons 3 niveaux (faible, moyen, élevé) pour chaque paramètre – $DR \in \{0.4, 0.6, 0.8\}$, $DF \in \{0.5, 0.7, 0.9\}$ et $DP \in \{0.3, 0.5, 0.7\}$ pour les expériences générées. Nous prenons compte les 27 combinaisons possibles. Pour chaque combinaison, 10 instances sont créées de façon aléatoire. Cela implique que chaque groupe de tests contient 270 instances.

Pour faciliter les notations, notons les problèmes par $Gn.\alpha\beta\gamma$, où n est le nombre des tâches non-fictives, $\alpha, \beta, \gamma \in \{1, 2, 3\}$ font référence au niveau choisi pour les paramètres de DR, DF, DP. Par exemple, notons $G15.132$ pour les instances avec 15 tâches, avec $DR = 0.4$ (niveau 1), $DF = 0.9$ (niveau 3) et $DP = 0.6$ (niveau 2).

Dans la section suivante, nous mettons en place les instances générées et comparons les méthodes présentées dans ce chapitre.

2.5.2 Méthodes exactes

Dans un premier temps, comparons les résultats obtenus par la TPM et la PPM. Commençons par $G10$, NC est fixé à 0.5 pour éviter une forte complexité du réseau. Les résultats trouvés par la TPM sont présentés dans le Tableau 2.3. Dans les colonnes "Nb Sol", nous donnons le nombre de problèmes résolus dans le sous-groupe $G15.\alpha\beta\gamma$. Les temps de calculs sont présentés dans les colonnes "Med, Min, Max", indiquant la valeur médiane, le minimum et le maximum du temps parmi les 10 instances d'un même sous-groupe. Le front de Pareto est trouvé pour tous les problèmes.

Tableau 2.3 – TPM – groupe G10, nombre de problèmes résolus et temps de calcul (en secondes)

G10-TPM		$\gamma = 1$				$\gamma = 2$				$\gamma = 3$			
	β	Nb Sol	Med	Min	Max	Nb Sol	Med	Min	Max	Nb Sol	Med	Min	Max
$\alpha = 1$	1	*10 / 10*	1.07	0.46	6.11	*10 / 10*	1.14	0.40	2.96	*10 / 10*	0.96	0.30	3.11
	2	*10 / 10*	1.49	0.43	5.34	*10 / 10*	1.74	0.82	7.11	*10 / 10*	1.71	0.72	3.57
	3	*10 / 10*	3.42	1.63	8.34	*10 / 10*	2.64	0.74	11.92	*10 / 10*	1.40	0.55	6.19
$\alpha = 2$	1	*10 / 10*	3.27	1.61	12.25	*10 / 10*	6.45	1.22	33.69	*10 / 10*	5.94	1.52	20.40
	2	*10 / 10*	9.30	6.41	44.67	*10 / 10*	7.94	3.98	26.86	*10 / 10*	11.10	4.47	36.93
	3	*10 / 10*	15.96	4.50	104.01	*10 / 10*	12.32	2.42	26.87	*10 / 10*	9.98	3.23	14.42
$\alpha = 3$	1	*10 / 10*	12.51	5.48	46.85	*10 / 10*	13.57	7.11	39.88	*10 / 10*	14.01	5.31	31.48
	2	*10 / 10*	8.72	3.53	13.90	*10 / 10*	20.41	8.65	82.03	*10 / 10*	20.90	2.20	73.74
	3	*10 / 10*	25.42	22.91	69.00	*10 / 10*	13.57	5.09	32.84	*10 / 10*	19.22	9.59	69.11

Le temps de calculs minimal et maximal sont respectivement 0.30s et 80.03s, toutes instances mélangées. Plus généralement, nous remarquons que lorsque α et β augmentent, les problèmes sont plus difficiles à résoudre. En ce qui concerne γ, il semble qu'il n'ait pas un effet régulier sur la difficulté des problème. Autrement dit, plus les ressources sont "partagées et demandées", plus

le problème devient difficile à résoudre. Certains contre-exemples peuvent être trouvés quand on passe de $\beta = 2$ à $\beta = 3$, surtout quand les valeurs de α, β et γ sont tous élevées (ex. G10.223 $\rightarrow$ G10.233). En effet, quand on augmente au maximum les trois paramètres, le nombre de solutions potentielles chute, et le nombre de sous-problèmes à résoudre diminue. Alors, temps de calcul diminue.

Tableau 2.4 – PPM – groupe G10, nombre de problèmes résolus et temps de calcul (en secondes)

G10-PPM		$\gamma = 1$				$\gamma = 2$				$\gamma = 3$			
	β	Nb Sol	Med	Min	Max	Nb Sol	Med	Min	Max	Nb Sol	Med	Min	Max
$\alpha = 1$	1	*10 / 10*	1.73	0.85	6.55	*10 / 10*	1.13	0.43	2.53	*10 / 10*	1.18	0.65	6.16
	2	*10 / 10*	1.75	0.81	3.82	*10 / 10*	2.05	0.74	3.55	*10 / 10*	2.72	0.80	8.16
	3	*10 / 10*	2.45	1.08	5.20	*10 / 10*	3.30	0.64	5.04	*10 / 10*	2.33	1.09	6.20
$\alpha = 2$	1	*10 / 10*	3.47	1.27	13.62	*10 / 10*	8.66	1.09	23.07	*10 / 10*	6.39	2.01	18.46
	2	*10 / 10*	12.27	3.80	42.02	*10 / 10*	8.49	2.78	40.23	*10 / 10*	8.82	3.03	29.70
	3	*10 / 10*	22.33	3.37	54.96	*10 / 10*	10.29	1.77	36.48	*10 / 10*	12.24	3.04	18.46
$\alpha = 3$	1	*10 / 10*	15.19	4.88	31.73	*10 / 10*	14.52	3.94	31.40	*10 / 10*	10.81	4.61	29.70
	2	*10 / 10*	14.77	4.40	19.59	*10 / 10*	20.66	5.54	75.62	*10 / 10*	12.77	2.48	61.55
	3	*10 / 10*	20.61	12.25	52.57	*10 / 10*	14.68	8.26	41.44	*10 / 10*	22.90	10.31	41.43

Les mêmes instances sont testées avec le PPM, dont les résultats sont donnés dans le Tableau 2.4, le nombre de répartition tenu en compte est $N_{PPM} = 4$. De même que la TPM, la PPM réussit à résoudre tous les problèmes testés. Le temps de calcul minimal et maximal sont respectivement 0.43s et 75.62s toutes instances mélangées. Les tendances observées sur le temps calcul trouvées avec la TPM sont confirmées avec la PPM.

Pour observer et mettre en évidence les temps de calculs de ces deux méthodes sur G10, nous présentons les valeurs Med, Min et Max pour chaque sous-groupe $\alpha\beta\gamma$ dans la Figure 2.12. Dans un premier temps, le graphe Med permet de mieux visualiser nos observations sur la difficulté des instances : le problème devient plus difficile à résoudre quand les ressources sont plus restreintes, alors que les due dates n'ont pas forcément d'influence régulière. En comparant les temps de calcul, on remarque que les performances globales de la TPM sont meilleures (voir graphe de la médiane). Cependant, vis-à-vis des temps de calcul minimal et maximal, la PPM est plus avantageuse dans la plupart des cas. Cela signifie que la PPM fonctionne mieux sur les cas extrêmes – temps de calcul très court ou bien très long, surtout lorsque la difficulté du problème devient plus élevée dont $\alpha = 2$ et $\alpha = 3$.

Considérons à présent un problème de plus grande taille avec les instances G15 et NC = 0.67. Les résultats de la TPM, de la PPM et leurs comparaisons sont montrés respectivement dans les Tableaux 2.5, 2.6 et la Figure 2.13. Dans nos expériences, le temps d'exécution est limité à 1800s, au-delà de ce seuil, le problème est considéré comme non-résolu.

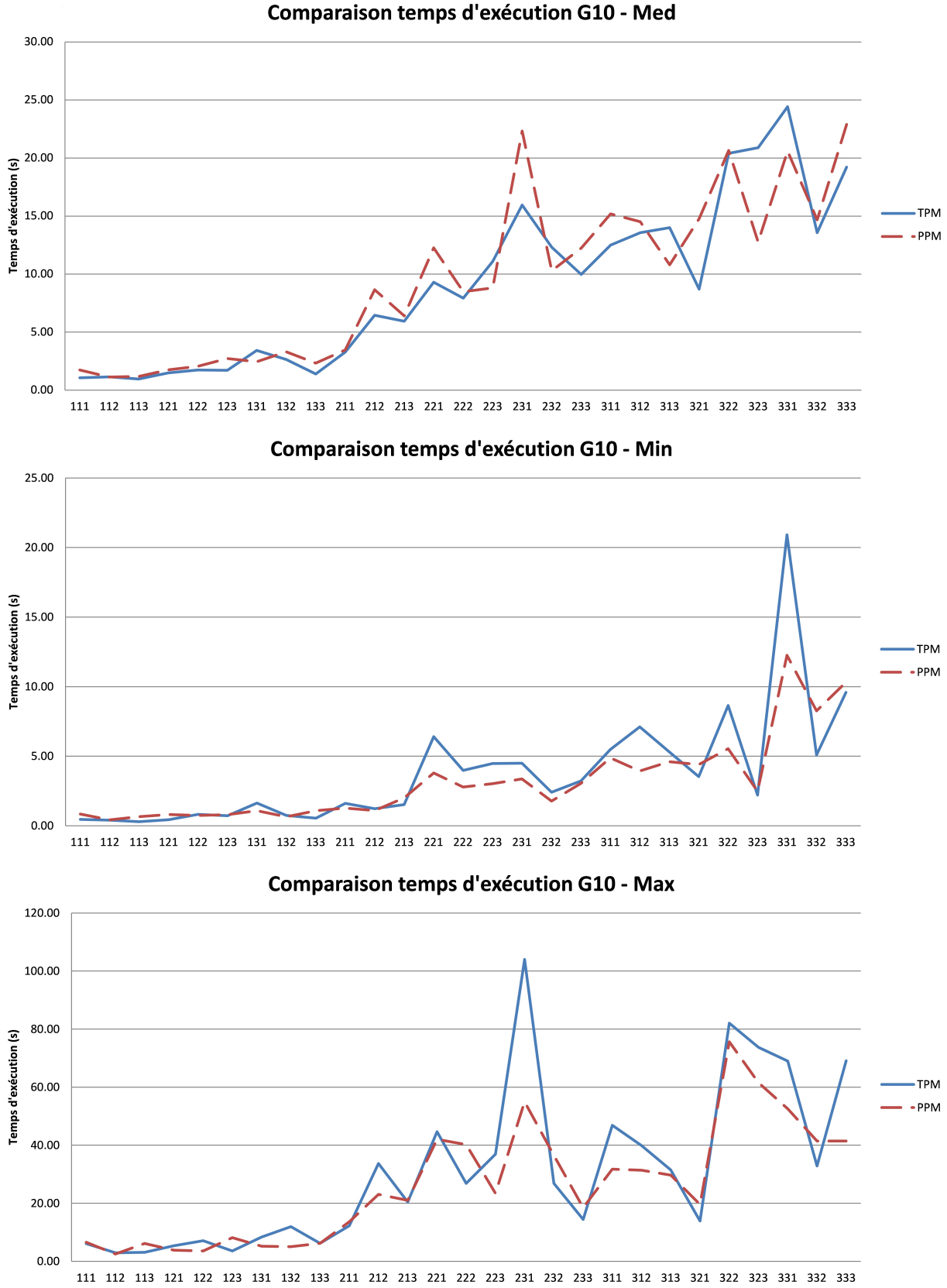

Figure 2.12 – Comparaison du temps d'exécution – TPM et PPM sur G10

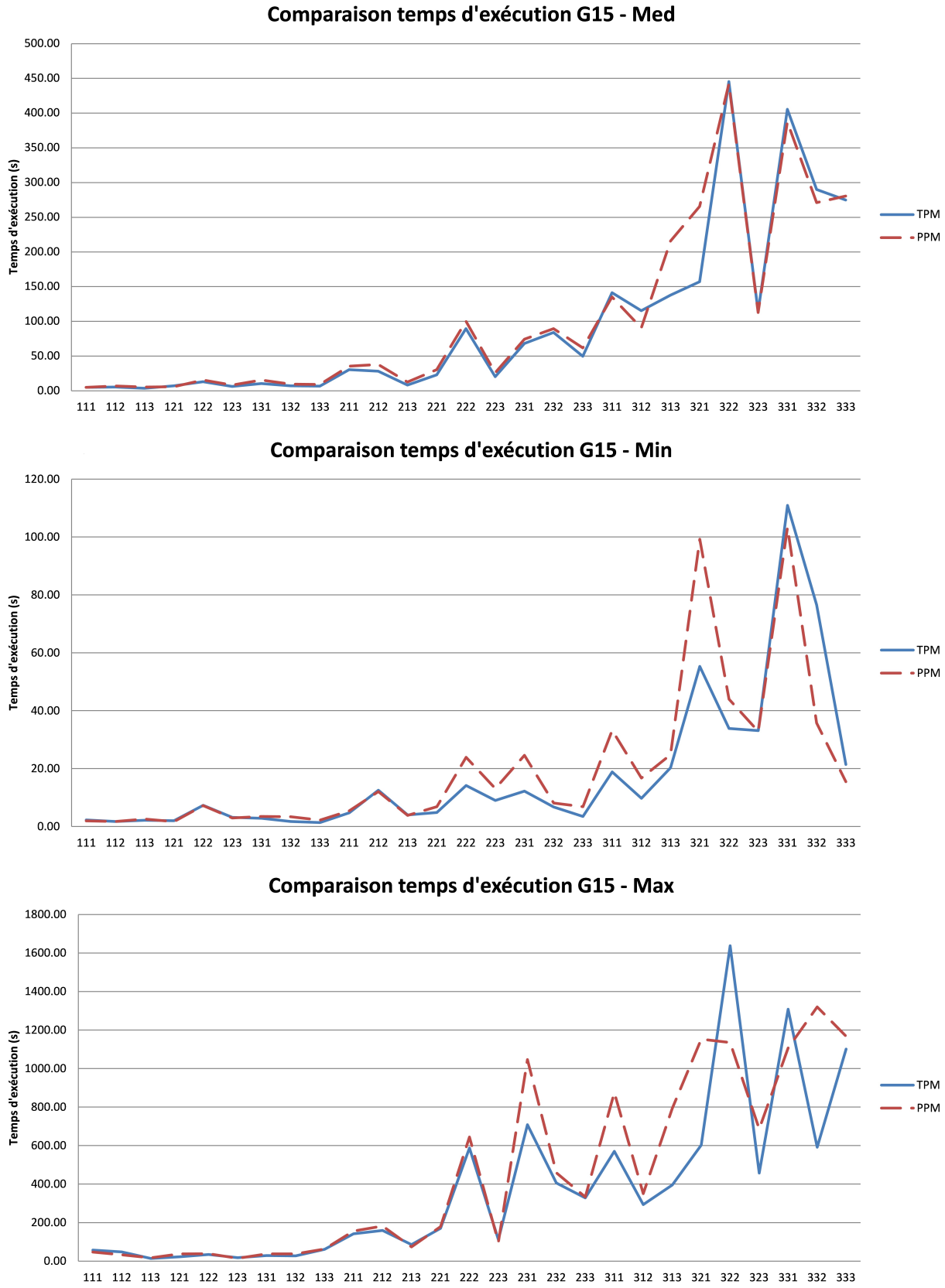

Figure 2.13 – Comparaison du temps d'exécution – TPM et PPM sur G15

A nouveau, les deux méthodes permettent de résoudre toutes les instances testées et confirment les observations générales sur les temps de calculs faites avec G10. D'ailleurs, les méthodes ont des comportements similaires sur plusieurs sous-groupes. Vis-à-vis de la médiane des temps de calcul, elles sont quasiment équivalentes pour tous les problèmes avec $\alpha = 1$ et $\alpha = 2$. Sur les sous-groupes avec $\alpha = 3$, le meilleur temps de calcul alterne entre les deux méthodes, mais leurs performances globales restent toujours proches. Il convient de noter que le temps maximum consommé par la PPM est plus avantageux lorsque les demandes de ressources sont très fréquentes ($\alpha = 3$). En effet, ces instances contiennent souvent moins de solutions, la PPM permet de réduire le nombre de sous-problèmes résolus.

Si l'on continue à augmenter la taille de prolème avec G20 et NC = 1.33, on voit dans les Tableaux 2.7 et 2.8, que l'on n'arrive pas à tout résoudre ni avec la TPM ni avec la PPM. Les deux méthodes montrent leur limites quand la taille des problèmes atteint 20 tâches. Ceci est dû à la capacité limitée du modèle mathématique : quand la résolution des problèmes avec des objectifs pondérés devient lourde, il devient impossible de trouver le front de Pareto en temps limité. Dans la Figure 2.14, nous mettons en évidence le nombre d'instances résolues au lieu d'illustrer les temps de calcul, car les deux méthodes ne permettent pas forcément de résoudre les mêmes instances. En considérant le nombre d'instances résolues, la TPM est plus performante que la PPM, surtout quand le problème est difficile à résoudre. Par exemple, dans les sous-groupes avec $\alpha = 3$, la TPM permet de résoudre plus d'instances, qui plus est avec des temps de calcul moins élevés. De même, notons que les remarques générales trouvées par les deux premiers groupes sont maintenues.

En conclusion, la TPM et la PPM permettent toutes deux de trouver le front de Pareto optimal. Vis-à-vis des problèmes testés, chacune est capable de résoudre toutes les instances G10 et G15, mais commence à avoir des difficultés lorsque le nombre de tâche atteint 20. Les deux méthodes partagent un même principe de résolution : créer et résoudre les sous problèmes avec des objectives pondérés. Néanmoins, les différentes stratégies d'exploration dans l'espace de recherche conduisent à des performances non-équivalentes. Dans toutes les groupes testés, le paramètres de parallélisme de la PPM (N_{PPM}) reste le même alors que le nombre de solutions potentielles augmente avec le nombre de tâches. Autrement dit, on peut considérer que $N_{PPM} = 4$ est "supérieur" au nombre de solutions attendues par G10, "juste" pour celui par G15 et "inférieur" à celui par G20. Remarquons que la meilleure performance de la PPM par rapport à la TPM est observée pour G15, où le paramètre choisi de façon adaptée. De ce fait, on peut conclure que la performance de la PPM dépend significativement du choix de son paramètre de parallélisme, alors que la TPM a une performance plus stable, et reste plus avantageuse que la PPM dans la plupart des instances considérés.

De plus, la difficulté des problèmes augmente non seulement avec le nombre de tâches, mais

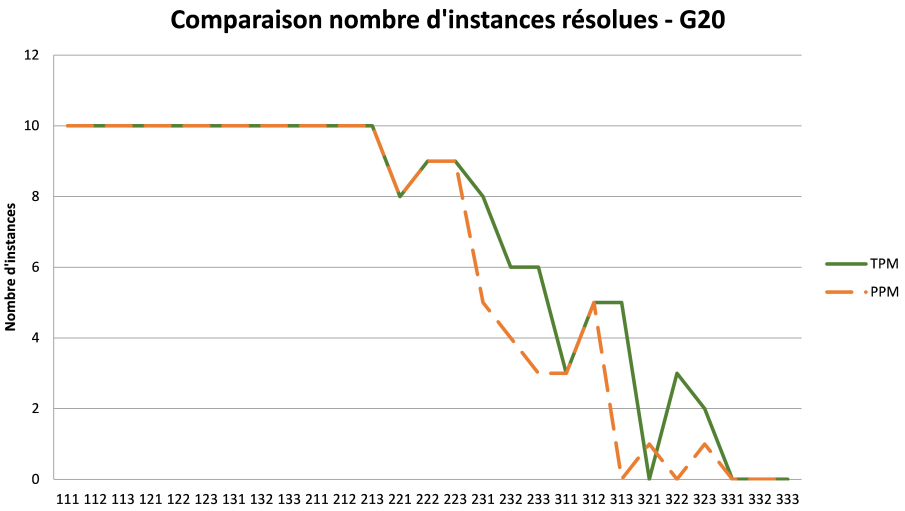

Figure 2.14 – Comparaison du nombre d'instances résolues

aussi avec la demande des ressources : le problème est plus difficile à résoudre lorsque la demande de ressources est élevée. Par exemple, pour G20, les deux méthodes réussissent à trouver le front de Pareto optimal pour les instances faciles. Cependant, quand la demande de ressources croit, une partie des instances deviennent rapidement impossible à résoudre en temps limité. Ceci explicite non-seulement la durée de la résolution exacte envisagée, mais aussi montre la dépendance entre la difficulté et les données des problèmes. C'est pourquoi nous mettrons en œuvre le NSGAII dans la section suivante, afin d'améliorer le temps d'exécution et de résoudre plus rapidement (surtout) les instances difficiles.

Tableau 2.5 – TPM – groupe G15, nombre de problèmes résolus et temps de calcul (en secondes)

G15		$\gamma = 1$				$\gamma = 2$				$\gamma = 3$			
	β	Nb Sol	Med	Min	Max	Nb Sol	Med	Min	Max	Nb Sol	Med	Min	Max
$\alpha = 1$	1	*10 / 10*	4.93	2.27	57.50	*10 / 10*	5.34	1.71	47.84	*10 / 10*	3.68	2.14	14.14
	2	*10 / 10*	7.08	1.97	22.40	*10 / 10*	13.03	7.31	34.50	*10 / 10*	6.27	3.13	17.97
	3	*10 / 10*	10.58	2.81	29.10	*10 / 10*	7.27	1.70	27.10	*10 / 10*	6.66	1.38	61.35
$\alpha = 2$	1	*10 / 10*	30.46	4.75	142.09	*10 / 10*	28.18	12.54	159.14	*10 / 10*	8.28	3.97	85.82
	2	*10 / 10*	22.88	4.86	170.76	*10 / 10*	89.55	14.14	587.66	*10 / 10*	20.45	8.98	112.21
	3	*10 / 10*	68.29	12.18	708.66	*10 / 10*	84.09	6.71	406.12	*10 / 10*	49.76	3.44	328.71
$\alpha = 3$	1	*10 / 10*	141.28	18.85	570.53	*10 / 10*	115.46	9.76	293.41	*10 / 10*	137.86	20.26	395.98
	2	*10 / 10*	157.27	55.30	602.32	*10 / 10*	445.40	33.85	1637.42	*10 / 10*	114.36	33.13	457.47
	3	*10 / 10*	405.53	110.93	1308.10	*10 / 10*	290.21	76.58	591.63	*10 / 10*	274.87	21.40	1100.91

Tableau 2.6 – PPM – groupe G15, nombre de problèmes résolus et temps de calcul (en secondes)

G15		$\gamma = 1$				$\gamma = 2$				$\gamma = 3$			
	β	Nb Sol	Med	Min	Max	Nb Sol	Med	Min	Max	Nb Sol	Med	Min	Max
$\alpha = 1$	1	*10 / 10*	4.91	1.94	47.30	*10 / 10*	6.98	1.72	32.97	*10 / 10*	5.48	2.58	17.20
	2	*10 / 10*	5.87	1.69	37.41	*10 / 10*	15.60	7.21	38.49	*10 / 10*	8.30	2.91	14.58
	3	*10 / 10*	15.48	3.47	37.39	*10 / 10*	9.64	3.35	37.85	*10 / 10*	9.14	2.22	62.62
$\alpha = 2$	1	*10 / 10*	35.56	5.42	156.00	*10 / 10*	37.95	12.05	182.51	*10 / 10*	12.64	3.86	74.45
	2	*10 / 10*	30.38	6.86	179.12	*10 / 10*	100.31	23.90	644.45	*10 / 10*	25.97	13.09	98.23
	3	*10 / 10*	74.39	24.61	1046.56	*10 / 10*	89.49	8.08	458.82	*10 / 10*	61.84	6.84	332.95
$\alpha = 3$	1	*10 / 10*	135.06	33.24	874.91	*10 / 10*	90.52	16.74	350.53	*10 / 10*	216.00	24.78	793.55
	2	*10 / 10*	265.75	99.23	1152.58	*10 / 10*	442.14	43.95	1134.83	*10 / 10*	112.78	32.83	686.39
	3	*10 / 10*	161.12	35.74	1319.80	*10 / 10*	271.12	35.74	1319.80	*10 / 10*	280.65	15.45	1167.86

Tableau 2.7 – TPM – groupe G20, nombre de problèmes résolus et temps de calcul (en secondes)

G20		$\gamma = 1$				$\gamma = 2$				$\gamma = 3$			
	β	Nb Sol	Med	Min	Max	Nb Sol	Med	Min	Max	Nb Sol	Med	Min	Max
$\alpha = 1$	1	*10 / 10*	30,98	10,94	56,21	*10 / 10*	18,72	5,93	114,27	*10 / 10*	37,68	5,61	77,78
	2	*10 / 10*	25,08	9,97	200,19	*10 / 10*	23,10	8,07	66,87	*10 / 10*	63,44	25,91	146,22
	3	*10 / 10*	50,29	12,87	137,66	*10 / 10*	24,99	13,67	98,43	*10 / 10*	212,01	40,49	979,54
$\alpha = 2$	1	*10 / 10*	267,62	69,79	771,39	*10 / 10*	188,82	30,15	622,83	*10 / 10*	293,93	89,20	1687,34
	2	8 / 10	580,77	266,85	909,38	9 / 10	185,92	103,60	1020,20	9 / 10	506,08	200,92	1059,30
	3	8 / 10	988,06	381,41	1309,33	6 / 10	746,07	381,04	1733,38	6 / 10	812,97	106,19	1778,09
$\alpha = 3$	1	3 / 10	754,76	485,76	1464,68	5 / 10	1074,83	66,68	1526,26	5 / 10	1196,36	720,46	1400,72
	2	0 / 10	-	-	-	3 / 10	926,33	647,53	1469,41	2 / 10	1066,89	777,04	1356,74
	3	0 / 10	-	-	-	0 / 10	-	-	-	0 / 10	-	-	-

Tableau 2.8 – PPM – groupe G20, nombre de problèmes résolus et temps de calcul (en secondes)

		$\gamma = 1$				$\gamma = 2$				$\gamma = 3$			
	β	Nb Sol	Med	Min	Max	Nb Sol	Med	Min	Max	Nb Sol	Med	Min	Max
$\alpha = 1$	1	*10 / 10*	37.99	6.96	60.41	*10 / 10*	27.97	4.36	92.80	*10 / 10*	34.62	14.73	59.55
	2	*10 / 10*	38.39	11.26	189.14	*10 / 10*	33.97	12.56	72.42	*10 / 10*	42.13	14.73	106.72
	3	*10 / 10*	73.56	30.87	235.94	*10 / 10*	35.24	13.35	195.36	*10 / 10*	158.66	23.56	738.97
$\alpha = 2$	1	*10 / 10*	474.67	89.38	1698.05	*10 / 10*	167.41	56.87	1060.97	*10 / 10*	318.87	63.08	1650.82
	2	8 / 10	832.48	262.87	1792.11	9 / 10	239.76	104.65	1146.64	9 / 10	319.87	129.69	1762.84
	3	5 / 10	1324.28	623.14	1785.19	4 / 10	886.91	754.44	1509.51	3 / 10	510.63	172.10	1170.44
$\alpha = 3$	1	3 / 10	1175.03	706.09	1378.38	5 / 10	620.14	139.88	1601.93	0 / 10	-	-	-
	2	1 / 10	1737.34	1737.34	1737.34	0 / 10	-	-	-	1 / 10	1704.13	1704.13	1704.13
	3	0 / 10	-	-	-	0 / 10	-	-	-	0 / 10	-	-	-

2.5.3 Méthode approchée

Puisque les méthodes exactes ne permettent de résoudre que des problèmes de petite taille, nous mettrons en place le NSGAII afin d'aborder les plus grands instances (voir le chapitre suivant). Cependant, avant de généraliser la méthode, nous comparons la NSGAII avec les méthodes exactes pour mesurer la qualité des fronts de Pareto approchés.

Trois critères principaux sont considérés lors de la comparaison – le temps de calcul, le métrique GD et IGD de la méthode approchée. Comme défini dans la section 1, le GD mesure la distance d'un front de Pareto approché au front de Pareto optimal, alors que l'IGD permet de mesurer la distance d'un front de Pareto optimal à un front approché. Lorsque l'IGD atteint 0, la solution approchée atteint l'optimum.

Prenons G15 comme des instances représentatives, où les méthodes exactes permettent de résoudre tous les problèmes testés mais avec du temps de calcul élevé pour certains. Pour le NSGAII, nous définissons la taille de population $N = 75$, le nombre de génération $n_G = 250$, le taux de croisement $c_r = 0.95$ et le taux de mutation $m_r = 0.15$. Les résultats sont présentés dans le Tableaux 2.9, où chaque case correspond au niveau moyen calculé à la base des 10 instances dans le sous-groupe.

Tableau 2.9 – NSGAII – métriques GD, IGD et temps d'exécution sur G15

G15		$\gamma = 1$			$\gamma = 2$			$\gamma = 3$		
	β	GD	IGD	Temps (s)	GD	IGD	Temps	GD	IGD	Temps (s)
$\alpha = 1$	1	**0.00**	**0.00**	10.64	**0.00**	**0.00**	10.85	0.07	0.07	10.53
	2	**0.00**	**0.00**	10.63	**0.00**	0.11	10.38	**0.00**	0.03	10.82
	3	**0.00**	0.23	10.35	0.12	0.11	10.57	**0.00**	0.06	10.37
$\alpha = 2$	1	0.07	0.30	10.65	0.11	0.09	10.82	**0.00**	0.27	10.88
	2	0.09	0.09	10.75	0.07	0.46	11.13	0.09	0.12	10.58
	3	0.16	0.41	10.67	0.22	0.34	10.44	0.18	0.24	10.74
$\alpha = 3$	1	0.15	0.35	11.16	0.03	0.38	10.87	0.55	0.58	11.15
	2	0.31	0.96	10.78	0.26	0.36	11.03	0.14	0.30	10.84
	3	0.26	0.69	11.04	0.28	0.42	10.45	0.24	0.21	10.52

En ce qui concerne le temps d'exécution, le NSGAII ne distingue pas parmi les sous-groupes et résout en 10.67s en moyenne, avec un minimum de 10.35s et un maximum de 11.16s. Alors que les méthodes exactes peuvent demander plus que 1100s pour résoudre certaines instances.

Au niveau de la qualité des solutions, le NSGAII arrive à trouver 100% les fronts de Pareto optimaux dans 3 cas parmi 27 sous-groupes, tout en sachant que chaque sous-groupe contient 10 instances. On remarque également que pour certains sous-groupes, le NSGAII trouve $GD = 0$ et

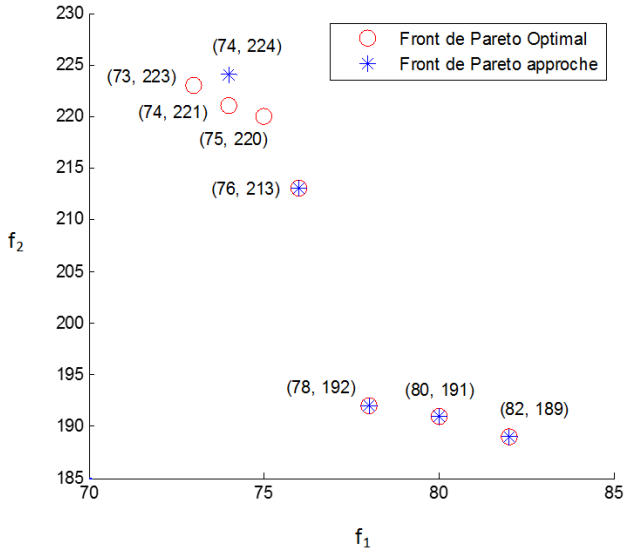

Figure 2.15 – Exemple avec GD = 0.60 et IGD = 1.22

$IGD > 0$. Cela signifie que toutes les solutions trouvées par le NSGAII sont des solutions Pareto-optimal. En revanche, toutes les solutions efficaces ne sont pas trouvées. Autrement dit, le NSGAII ne trouve qu'un front de Pareto "partiel".

Rappelons que notre espace de solution est un espace discret. Comme référence, le nombre de solutions en moyenne du G15 est 3.33. Lorsque l'on décale 1 par rapport à une solution efficace, les valeurs de GD/IGD augmentent potentiellement par 0.3 pour l'instance impliqué, et alors 0.03 pour la valeur du sous-groupe. Par conséquent, la qualité des solutions de certains groupes est bonne malgré des valeurs de GD et IGD non nulles. Par exemple, considérant le groupe G15.123, le NSGAII trouve le front de Pareto optimal sur 9 instances parmi les 10. Un décalage de 2 sur f_1 par rapport une solution efficace est détecté sur la seule instance restante.

Dans la Figure 2.15, nous donnons un exemple typique parmi les instances où 0 < GD < IGD. Comme on peut le voir, 4 parmi les 5 solutions trouvées par le NSGAII font partie du front de Pareto Optimal. La solution (74, 224) est entourée par (73, 223), (74, 221) et (75, 220). Cependant, elle ne parvient pas à sortir de son local et converger vers ces solutions de Pareto. Vis-à-vis de cette situation, on envisage à mettre en place les techniques d'amélioration pour assurer la qualité des solutions, et qui fera en partie l'objet de chapitre suivant.

2.5.4 Conclusion

Dans ce chapitre, on s'est concentré sur un problème bi-objectif optimisant le makespan et le retard total des tâches. Deux formulations mathématiques sont présentés pour modéliser le problème – le modèle 1 est plus intuitif alors que le modèle 2 est plus performant.

C'est pourquoi nous intégrons le modèle 2 dans la TPM et la PPM, les deux méthodes exactes mises en œuvre pour résoudre le problème bi-objectif. Néanmoins, les approches exactes ne permettent de résoudre que les problèmes avec des instances de petite taille. Selon les tests réalisés, chacune des méthodes peut résoudre les problèmes avec 10 tâches (G10) et 15 tâches (G15) avec le temps de calcul limité à 1800s. Lorsque l'on considère G20, les deux méthodes ne sont plus capables de résoudre les instances difficiles.

Par conséquent, le NSGAII est mis en place comme méthode approchée. Pour tester son efficacité, on a pris G15 comme exemple et les solutions sont mesurées par le temps d'exécution ainsi que les métriques GD et IGD. Le temps de calcul est sans doute un avantage. En ce qui concerne la qualité des fronts de Pareto approchée, le NSGAII propose des fronts de bonne qualité voire des fronts optimaux sur la plupart des instances. Cependant, sur certains problèmes, la performance du NSGAII est limitée. Pour cette raison, des voies d'amélioration par des schémas de constructions particuliers des hybridations ou autres sont explorées. Une partie des résultats seront présentés dans les chapitres dans la suite de ce document.

Chapitre 3

Un problème multi-objectif : méthodes approchées et recherche locale

Dans le chapitre précédent, deux objectifs ont été pris en compte – le makespan et le retard total des tâches. Ces deux objectifs s'intéressent à la productivité ainsi qu'à la qualité du service. Dans ce chaptire, nous considérons un troisième critère portant sur les ressources – l'équilibrage d'allocation des ressources au cours du temps. Ce problème multi-objectif est introduit dans la section 3.1. Dans un deuxième temps, le NSGAII est adapté pour résoudre ce dernier (section 3.2.1). Enfin nous implémentons le NSGAIII [18] dans la section 3.2.2. A la fin du chapitre précédent, nous avons fait une remarque sur le potentiel d'amélioration sur les fronts de Pareto approchés. Pour remédier à ce problème, une recherche locale avec la technique de Mapping, proposée par Autuori et al. (2016) [4], est ensuite mise en œuvre afin d'améliorer la qualité des solutions (section 3.3).

3.1 Introduction d'un nouveau critère

La gestion des ressources au cours du temps distingue le RCPSP des autres problèmes d'ordonnancement. Cependant, elles ne sont évaluées que dans peu d'études. Nous prenons en compte l'équilibrage de charge comme un troisième objectif et introduisons ce critère dans la littérature du RCPSP pour la première fois.

Ouazene et al. ont proposé un modèle d'équilibrage de charge sur un problème de machines parallèles [53]. Une nouvelle formulation est proposée dans leur étude permettant d'évaluer l'utilisation des ressources. L'équilibrage de charge était alors réalisé à partir de la différence entre l'utilisation maximale et minimale entre toutes les machines. Il est ensuite prouvé que cette formulation est plus efficace que des plus "traditionnelles" où seule l'utilisation maximale est minimisée.

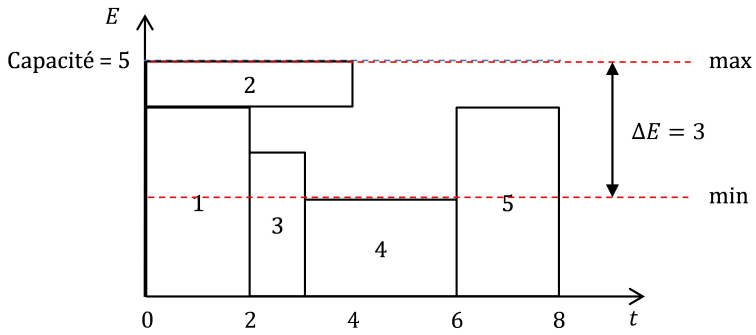

Figure 3.1 – Solution avec $f_1 = 8, f_2 = 2, f_3 = 3$

Nous reprenons donc dans ce travail l'idée de "minimiser un écart" et proposons de minimiser la différence entre la consommation maximale et minimale des ressources :

$$minimiser \quad \Delta E = \sum_{r \in R} (E_r^{\max} - E_r^{\min}) \tag{3.1}$$

où $E_r^{\max}$ et $E_r^{\min}$ sont respectivement la demande maximale et minimale de la ressource $r \in R$ sur tout l'horizon. Reprenons le modèle mathématique présenté et les notations utilisées dans la section 2.2.2, les contraintes de ressources peuvent être écrites comme suit :

$$E_r^{\max} \geq \sum_{i=1}^{n} q_{ir}(s_{it} - f_{it}) \quad \forall t \in T, \forall r \in R \tag{3.2}$$

$$E_r^{\min} \leq \sum_{i=1}^{n} q_{ir}(s_{it} - f_{it}) + M * f_{(n+1)t} \quad \forall t \in T, \forall r \in R \tag{3.3}$$

Dans la contrainte 3.3, le nombre M est introduit afin que la contrainte soit toujours vérifiée une fois que la tâche $n + 1$ est terminée.

En effet, ce critère permet d'évaluer le lissage de l'utilisation des ressources et de vérifier que l'allocation des ressources est bien équilibrée au cours du temps. Notons par la suite, cet objectif par f_3. La Figure 3.1 propose une illustration de ce problème sur l'exemple déjà utilisé dans le chapitre 1, où f_3 est mis en évidence.

3.2 Méthodes approchées

Il a été mis en évidence expérimentalement que les méthodes exactes ne permettent de résoudre qu'une partie des petites instances pour le problème bi-objectif. Quant au problème multi-objectif, la recherche du front de Pareto optimal est encore plus difficile. De plus, la résolution exacte des problèmes avec 3 objectifs demande beaucoup plus d'étapes que celle des problèmes bi-objectif [58].

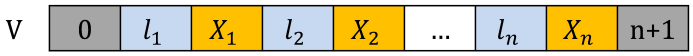

Figure 3.2 – Représentation générale du chromosome – problème multi-objectif

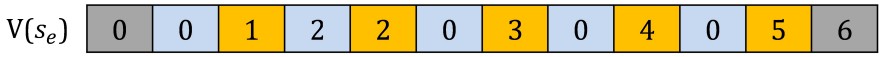

Figure 3.3 – Exemple du chromosome pour le problème multi-objectif avec $l_2 = 2$

C'est pourquoi nous avons choisi de résoudre notre problème multi-objectif à l'aide de méthodes approchées. Le NSGAII est d'abord adapté pour tenir compte de ce nouveau critère. Le NSGAIII, récemment proposé par Deb et Jain [18], est aussi implémenté dans ce travail.

3.2.1 Adaptation du NSGAII

3.2.1.1 Codage du chromosome

Rappelons que pour le problème bi-objectif, les chromosomes sont représentés par les séquences de priorité. Après avoir intégré f_3, il est plus intéressant de chercher les solutions qui permettent d'obtenir un meilleur lissage de l'utilisation des ressources. Pour ce faire, nous introduisons un temps de latence l_i pour chaque tâche $i \in J$, un gène est donc défini comme un couple (l_i, X_i). Ainsi, le chromosome peut s'écrire comme $V =< 0, (l_i, X_i), n+1 >$ (voir la Figure 3.2). Reprenons l'exemple utilisé précédemment (Section 2.4, Figure 2.9) : si on ajoute un temps de latence pour $l_2 = 2$ pour la tâche 2, nous obtenons le chromosome illustré en Figure 3.3, et la nouvelle solution est représentée dans la Figure 3.4.

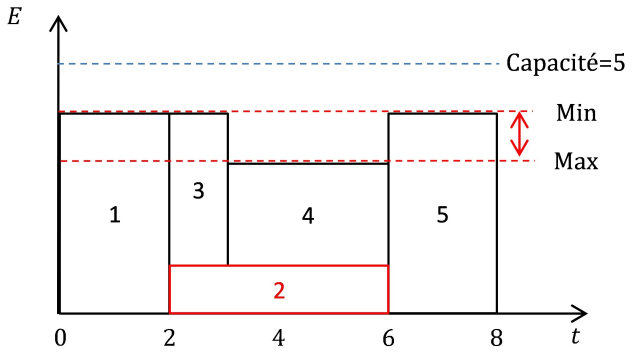

Figure 3.4 – Solution avec $f_1 = 8, f_2 = 4, f_3 = 1$

En décalant la tâche 2, on obtient une solution où l'utilisation des ressources est mieux équilibrée. Cependant, comme cette tâche est obligée d'attendre, elle génère un retard plus élevé.

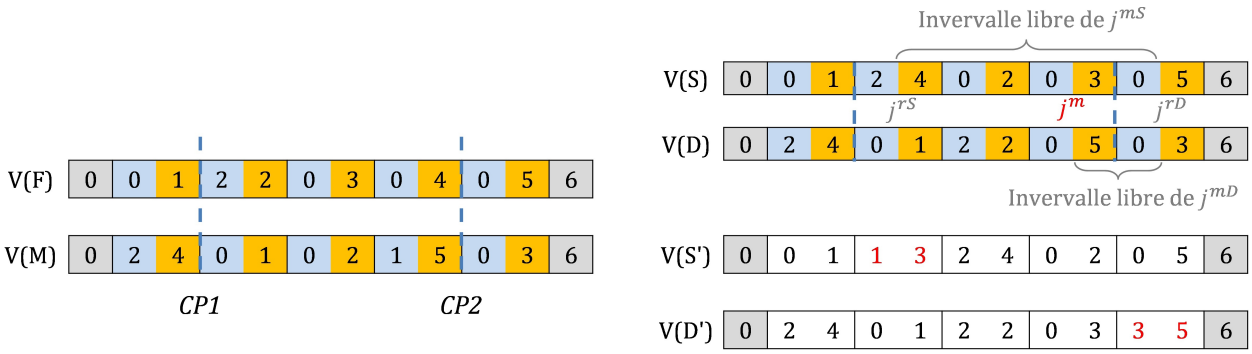

Figure 3.5 – Croisement et mutation – problème multi-objectif

Par conséquent, lorsque l'on crée la population initiale, plusieurs paramètres supplémentaires doivent être pris en compte :

- P_l : probabilité d'avoir un temps de latence effectif (non-nulle) pour une position donnée
- $l^{\max}$: Valeur maximale des temps de latence positives

3.2.1.2 Croisement et mutation

Le croisement et la mutation doivent être adaptés au codage des chromosome décrit en Section 3.2.1.1. On conserve les mouvements de croisement à un point et à deux points, et l'insertion pour la mutation (voir sections 2.4.3.1 et 2.4.3.2).

Si une tâche est choisie comme j^m, son temps de latence est réinitialisé à 0 s'il n'était pas nul. Sinon, on choisit une valeur aléatoire dans $[1, l^{\max}]$ pour avoir un temps de latence positif, comme on peut le voir dans la Figure 3.5.

3.2.1.3 Sélection

La sélection est réalisée de la même façon que dans le chapitre le chapitre précédent. Comme expliqué à la fin de la section 2.4.4, lors du tri d'un ensemble de solutions non-dominées dans un problème bi-objectif, l'ordre croissant sur un objectif donne l'ordre décroissant sur le deuxième objectif. Pour cette raison, le calcul des *crowding distances* est simplifié. Cependant, cette propriété n'est plus valide dans le cas multi-objectif (plus de deux objectifs). Par conséquent, lors du calcul de la *crowding distance*, il est impératif de considérer les trois objectif un par un. L'algorithme de calcul reste le même comme que celui décrit dans l'Algorithme 1.

3.2.2 Adaptation du NSGAIII

Le NSGAIII a été proposé par Deb et Jain [18] en 2014 comme une version améliorée du NSGAII. Il est destiné spécialement aux problèmes multi-objectifs et propose une nouvelle procédure de sélection. Le NSGAIII reprend les principes du NSGAII, avec en particulière le schéma général, le croisement, la mutation et la sélection basée sur la dominance de Pareto.

Nous allons, ci-après, expliquer la principale différence entre le NSGAII et le NSGAIII, qui concerne la phase de sélection parmi les solutions qui sont Pareto-équivalentes. Pour ce faire, le NSGAIII propose d'utiliser des *hyperplans et points de référence* pour évaluer la distribution des solutions sur un même front non-dominé. Dans un premier temps, un hyperplan normalisé est créé selon les solutions déjà choisies. Un ensemble de points de référence uniformément repartis sont ensuite identifiés. Pour chaque solution sur le front d'intérêt, on associe un point de référence. La répartition des solutions est identifiée à partir des points de référence.

3.2.2.1 Hyperplan et points de référence

Afin d'évaluer la répartition des solutions, nous identifions d'abord les valeurs Ideal et Nadir sur chaque objectif, pour l'ensemble de la population. Notons I le point Ideal et s_1, s_2, s_3 les solutions proposant les valeurs maximales sur f_1, f_2 et f_3. Nous pouvons définir l'hyperplan H passant par s_1, s_2 et s_3. Notons $(d_1, 0, 0)$, $(0, d_2, 0)$, $(0, 0, d_3)$ les points d'intersection de H avec chacun des axes, F_e le front d'intérêt. Une solutions $s \in S = \{F_1, F_2, \dots F_e\}$ peut être normalisée comme le suivant :

$$f_k'(s) = \frac{f_k(s) - f_k(I)}{d_k}, \forall k = \{1, 2, 3\} \tag{3.4}$$

Ainsi, l'ensemble de solutions normalisées peut être déterminé et notons-le par S'. Avec le même principe, on obtient les intersections normalisées :

$$d_k' = 1, \quad \forall k = 1, 2, 3 \tag{3.5}$$

Par conséquent, nous pouvons identifier l'hyperplan normalisé, tel que :

$$f_1' + f_2' + f_3' = 1 \tag{3.6}$$

qui est borné par les droites $f_1' + f_2' = 1$, $f_1' + f_3' = 1$ et $f_2' + f_3' = 1$, comme illustré dans la Figure 3.7. A l'intérieur de cet espace borné, un ensemble de points de références Z sont uniformément positionnés. Si n_S divisions sont considérées sur chaque arrête, nous obtenons $C_{3+n_S-1}^{n_S}$ points de référence. Dans la Figure 3.6, un simple exemple est donné avec $n_S = 3$.

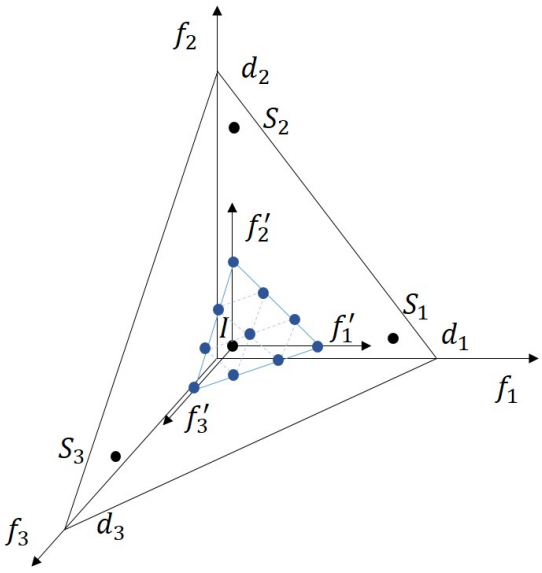

Figure 3.6 – Exemple avec $n_S = 3$ et 10 points de référence

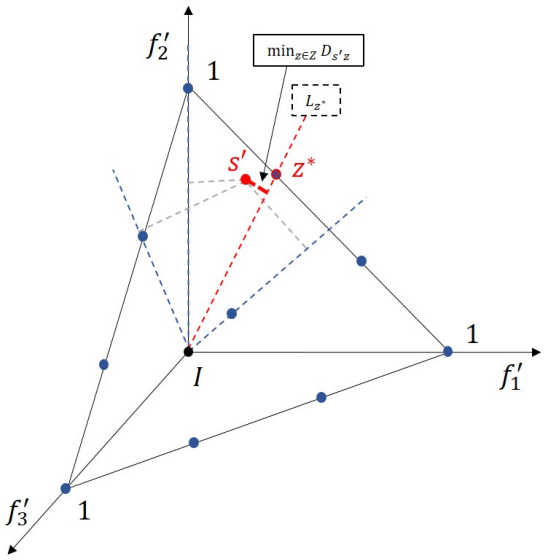

Figure 3.7 – Choisir la droite de référence

3.2.2.2 Solutions et points de référence

Pour chaque point $z \in Z$, on peut tracer une ligne de référence correspondante en reliant z et le point Ideal I. Notons L, l'ensemble des droites de référence et L_z la droite définie par $z \in Z$.

Afin d'établir les relations entre les points de référence $z \in Z$ et les solutions $s \in S$, nous calculons pour chaque solution normalisée $s' \in S'$, la distance de s' à chacune des droites de référence. Notons $D_{s'z}$ la distance de s' à L_z. La solution s est associée au point de référence z^* qui minimise la distance $D_{s'z}$ de la droite L_z à s'. Autrement dit :

$$D_{s'z^*} = \min_{z \in Z} D_{s'z} \implies s \sim z^*, \quad \forall s' \in S' \tag{3.7}$$

où $s \sim z^*$ représente la relation d'association entre la solution non-normalisée s et le point de référence. On donne la Figure 3.7 qui montre un exemple d'association : L_z* propose la plus petite distance perpendiculaire depuis s' et z^* est ensuite associée avec s.

3.2.2.3 Sélection à l'intérieur d'un front

Une fois toutes les solutions associées à un point de référence, la distribution des solutions est évaluée en ces seuls points de références. Rappelons que F_e est le front à étudier. Autrement dit, les solutions appartenant aux fronts avant F_e sont déjà choisies par le programme. Pour chaque point de référence $z \in Z$, notons par N_z le nombre de solutions $s \in \{F_1, F_2, \dots F_{e-1}\}$ associées à z. Afin de rendre la distribution des solutions plus homogène, les points de références ayant la

plus faible valeur de N_z sont considérés les moins visités. Par conséquent, les solutions z dont N_z est grand sont en priorité dans la procédure de sélection. Nontons z^* le point de référence associé au plus faible nombre de solutions :

$$z^* \mid N_{z^*} = \min_{z \in Z} N_z, \quad z \in Z \tag{3.8}$$

Ceci signifie que z^* se situe dans la zone la moins visitée. Pour cette raison, nous allons favoriser les solutions associées à z^*. Dans le cas où plusieurs points de référence vérifient l'équation (3.8), un point de référence z^* est aléatoirement choisi parmi ces derniers comme celui d'intérêt. Puis une solution qui se situe sur F_e associée est aléatoirement choisie pour la prochaine génération. Par conséquent, N_{z^*} est mis à jour et incrémenté de 1. Dans le cas où z^* n'est associé à aucune solution dans F_e, il est exclu et ne sera plus considéré par la suite. Avec les informations à jour, nous répétons ces étapes "déterminer $z^* \rightarrow$ choisir solution $\rightarrow$ mettre à jours N_{z^*}". Quand le nombre de solutions choisies sur F_e atteint le nombre attendu, la procédure est arrêtée.

Les algorithmes NSGAII et NSGAIII sont des métaheuristiques, elles fournissent donc des résultats acceptables pour des problèmes de grande taille. Néanmoins, la contrepartie de cette efficacité est le risque de blocage près d'un optimum local qui limite alors la diversification de la recherche.

3.3 Recherche locale avec la méthode de Mapping

Les algorithmes NSGAII et NSGAIII sont des métaheurstiques, elles fournissent donc des résultats acceptables pour des problèmes de grandes tailles. Néanmoins, la contrepartie de cette efficacité est le risque de blocage près d'un optimum local qui limite alors la diversification de la recherche. La difficulté principale consiste alors à la qualité des solutions.

Pour remédier à ce problème, deux approches sont souvent utilisées pour améliorer la qualité, de manière intelligente. Tout d'abord, il est possible de créer les individus en tenant en compte des caractéristiques propres au problème étudié. Ceci implique qu'il est nécessaire de développer des opérateurs (croisement, mutation, etc.) dédiés spécialement au problème concerné. Afin d'éviter de tomber dans un minimum local, une recherche locale peut également être mise en place afin d'explorer les voisinages des solutions déjà trouvées. Dans ce travail, nous avons choisi les opérateurs d'offspring génériques (croisement one-point et two-point, mutation type insertion) et puis nous améliorons les résultats à l'aide d'une recherche locale.

Avant de procéder à la recherche locale, trois questions peuvent se monter et nous attirent l'attention :

- Quand appliquer la recherche locale ?

- Sur quelles solutions est-il préférable de l'appliquer ?
- Quel est le technique utilisé pour explorer le voisinage d'une solution ?

Vis à vis de ces question, nous présentons dans un premier temps la méthode pour l'exploration du voisinage dans la section 3.2.1. Puis nous détaillerons dans la section 3.2.2 comment nous avons sélectionné les solutions auxquelles nous allons appliquer la recherche. Enfin nous aborderons la question de fréquence de la recherche locale dans la section 3.2.3.

3.3.1 Technique de la recherche locale

Dans notre travail, la technique de base utilisée pour explorer les voisinages des solutions est l'insertion. Pour ce faire, nous faisons référence à l'opérateur de mutation utilisé dans la section 2.4.3.2. Nous rappelons que les deux positions essentielles dans la mutation sont la tâche mutante j^m et la tâche de référence j^r. Lors d'une mutation, j^m et j^r sont choisies aléatoirement tandis que dans la recherche locale, toutes les positions possibles de j^m et j^r sont potentiellement prises en compte. Dans l'algorithme 2, nous donnons un résumé sur l'exploitation dans le voisinage d'une solution donnée s.

L'objectif de cette procédure est de trouver, à partir de s, une solutions $\tilde{s}$, qui domine s ou non-dominée par s. Pour ce faire, une tâche d'insertion j^{in} est choisie aléatoirement parmi toutes les tâches non étudiées ; notons $p(j^{in})$ sa position et Q l'ensemble de positions non-étudiées. A l'état initial, Q est l'ensemble de toutes les positions non-fictives. Les tâches sont étudiées au fur à mesure, et les positions correspondantes sont alors enlevées – dès que j^{in} est choisie, elle est exclue de l'ensemble non étudié. Nous cherchons ensuite la fenêtre de liberté de j^{in}, définie par les positions de sa dernière finie prédécesseur j^{dp} et sa première commencée successeur j^{ps}. Ainsi, nous pouvons choisir la tâche de référence, telle que $p(j^r) \in [p(j^{dp}) + 1, p(j^{ps}) - 1]$.

Rappelons que dans la mutation, j^r est choisie aléatoirement dans l'intervalle valide ; alors que dans la recherche locale, nous considérons potentiellement toutes les positions possibles. Ensuite nous commençons par la première position $p(j^{dp}) + 1$ et réalisons la procédure d'insertion. Si la solution trouvée $\tilde{s}$ vérifie les conditions d'acceptations, elle est considérée comme conforme et on arrête la procédure. Sinon, nous continuons sur $p(j^r) = p(j^{dp} + 2)$ et mettons à jour la solution $\tilde{s}$.

Après avoir testé toutes les positions dans $[p(j^{dp}) + 1, p(j^{ps}) - 1]$, si aucune solution trouvée n'améliore la solution de départ, nous considérons une autre tâche d'insertion et mettons à jour j^{in}. Une nouvelle fois, nous cherchons j^{dp} et j^{ps} pour j^{in}, et testons ensuite toutes les positions possibles de j^r.

La procédure prend fin dès que l'on trouve une solution $\tilde{s}$ qui est au minimum non-dominée par s.Cette recherche est considérée "réussie" si elle produit une solution $\tilde{s}$ conforme, sinon elle est

Algorithm 2 Recherche locale – voisinage de s

Input : $s, V(s)$ – une solution et son chromosome
Q(s) – ensemble de positions disponibles pour effectuer l'insertion
rand(Q(s)) – fonction qui permet de choisir aléatoirement un élément dans l'ensemble *Q(s)*
Output : $\tilde{s}$ – solution tenue en compte lors de la recherche locale
Variables : j^{in} – tâche choisie pour effectuer l'insertion, équivalente de la tâche de mutation
j^r – tâche de référence
j^{dp} – tâche dernièrement finie prédécesseur de j^{in}
j^{ps} – tâche premièrement commencée successeur de j^{in}
found – variable booléenne, qui prend la valeur *vrai* si $\tilde{s}$ est trouvée, et *faux* sinon

Début
Définir *found* = *faux*
Tant que (*found* = *faux*) et ($Q(s) \neq \varnothing$) **faire**
$p(j^{in}) = rand(Q(s))$
$Q(s) = Q(s) \backslash \{p(j^{in})\}$
Trouver j^{dp} et j^{ps}
$i = p(j^{dp})/2 + 1$
Tant que (*found* = *faux*) et ($2i < p(j^{ps})$) **faire**
$j^r = V(s, 2i)$
Procéder l'insertion (voir section 2.3) avec j^{in} et j^r et obtenir $\tilde{s}$
Si ($\tilde{s} \prec_P s$) ou $[(\tilde{s} \not\prec_P s)$ et $(s \not\prec_P \tilde{s})]$ **alors**
found = *vrai*
else
$i = i + 1$
Fin si
Fin tant que
Fin tant que
Si *found* = *vrai* **alors**
Retourner $\tilde{s}$
else
Retourne $\varnothing$
Fin si
Fin

"échouée". De ce fait, cette procédure de l'exploitation du voisinage peut être considérée comme une "pseudo-mutation".

Mise à part l'"insertion", plusieurs techniques de recherche locale sont aussi fréquemment trouvées dans la littérature comme le "switch" et "swap". Nous avons choisi l'insertion dans ce travail, qui partage la même racine de la mutation. Ainsi, toutes les solutions issues de la recherche locale sont également théoriquement atteignables par la mutation.

3.3.2 Méthode de Mapping

Dans la section précédente, nous nous sommes concentrés sur la technique de recherche locale sur une *solution donnée*. Cependant, avant cette étape, nous devons choisir les solutions sur lesquelles on va appliquer la recherche locale. Et ces choix ont une influence importante sur la qualité des résultats.

Lorsque l'on choisit les solutions sur lesquelles appliquer la recherche locale, la question qui se pose est : quelles sont les solutions dont les voisinages sont intéressants à explorer ? La réponse traditionnelle à cette question dépend, dans la plupart des cas, du hasard (par exemple, une solution est choisie avec une certaine probabilité) ou des valeurs objectives (par exemple, les solutions sur le front 1 sont prises).

Dans cette, nous utilisons la méthode de Mapping ("Mapping Method (MaM)"), proposé par Autuori et al. (2016) [4], pour répondre à cette question. Cette méthode analyse les chromosomes des solutions, les projette sur une carte ("*Map*"), puis les regroupe selon leur structure. La MaM [4] permet de transformer l'espace des solutions mutli-dimensionnel en une Map uni-dimensionnel, à l'aide d'une *bijection* entre les solutions et leurs positions sur la Map. Pour ce faire, la MaM s'appuie sur les structures des individus à l'aide de la distance de *Hamming*.

Cette technique a récemment été élaborée et appliquée par Autuori, Hnaien et Yalaoui [4] pour résoudre un problème de Job-Shop flexible ("Flexible Job-Shop Problem (FJSP)"), où 5 stratégies d'exploration sont proposées. Dans leurs travaux, le NSGAII est pris comme exemple de méthodes approchées, couplé avec la MaM pour une meilleure exploration de l'espace des solutions. Nous adaptons dans ce chapitre ces stratégies à notre problème, et l'élargissons ensuite en considérant la MaM avec plusieurs règles de dominance dans le chapitre suivant.

Pour mieux expliquer l'implémentation de la MaM, nous résumons les démarches par plusieurs étapes principales, qui correspondent aux différentes sous-sections :

- Étape 1 : Décodage les chromosomes en représentation binaire (section 3.3.2.1)
- Étape 2 : Choix d'une solution de référence (section 3.3.3) et calcul des *Hamming vecteurs* et des *Hamming distances* pour chaque solution (section 3.3.2.2)
- Étape 3 : A partir de la *fonction de Mapping bijective* (section 3.3.2.3), localisation des solutions sur la Map
- Étape 4 : Identification des différentes zones et calcul de la distribution des solutions, sélection des solutions à améliorer par la recherche locale (section 3.3.2.4)

Pour une meilleure compréhension, nous donnerons un exemple numérique dans la Figure 3.8, dont les éléments seront expliqués au fur et à mesure par la suite.

3.3.2.1 Décodage du chromosome

Pendant le décodage d'une solution, nous devons garantir la *bijection* entre le chromosome et son décodage. Il est aussi important que le vecteur de Hamming (voir section 3.3.2.2) puisse représenter les mouvements d'une solution s par rapport à la solution de référence s^r. Pour cette raison, les gènes X et les gènes l sont gérées de manières différentes. De plus, les tâches fictives

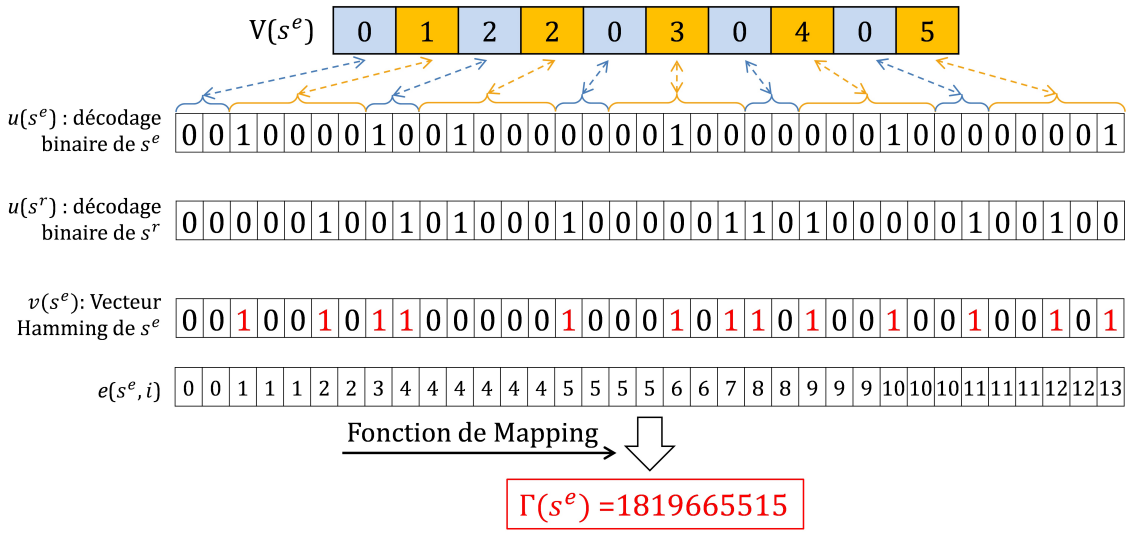

Figure 3.8 – Méthode de Mapping – un exemple de décodage et calculs

0 et $n + 1$ ne sont pas prises en compte car elles restent toujours aux mêmes positions pour toutes les solutions.

Les gènes X concernent l'affectation "position – tâche". Ainsi, nous proposons pour chaque gène X, un vecteur binaire de taille n, où le $b^{\text{ième}}$ bit prend la valeur 1 si $p(a) = b$; c'est à dire si la tâche a est affectée à la $b^{\text{ième}}$ position dans le chromosome. On note le décodage d'une solution donnée s par $u(s)$. Prenons l'exemple de s_e avec 5 tâches non-fictives (aussi utilisé dans la section 1 et 2), comme montré dans la Figure 3.8. La première tâche choisie dans $V(s^e)$ est la tâche 1, alors le décodage correspondant dans $u(s^e)$ est écrit par $(1.0.0.0.0)$. Par conséquent, les gènes X sont transformés en $n * n = 25$ bits dans le vecteur binaire décodé.

Pour les gènes l, un décodage direct en nombres binaires est utilisé, comme dans les travaux de Autuori et al. [4]. Il est d'ailleurs important de décoder tous les gènes l de même taille. Dans ce but, la plus grande valeur de la latence doit être identifiée avant tout décodage, ce qui est égale 2 dans notre exemple. Tous les gènes l sont donc décodés en 2 nombres binaires, comme on peut le voir dans la Figure 3.8.

De ce fait, le décodage final de $V(s^e)$, noté par $u(s^e)$, comprend 35 nombres binaires, dont 25 pour les gènes X et 10 pour les gènes l.

3.3.2.2 Distance de Hamming

La méthode de Mapping propose de créer une "Map", où chaque solution peut trouver une position *unique* selon son chromosome. Autrement dit, elle cherche à créer une *relation bijective* entre les solutions et leurs positions sur la Map. A cet effet, nous avons besoin de la notion de "distance

de Hamming". Pour deux vecteurs $u_1 = (u_1^1, u_1^2, \ldots, u_1^Z)$ et $u_2 = (u_2^1, u_2^2, \ldots, u_2^Z)$ de la même taille Z, définissons le "vecteur de Hamming" une série de nombres binaires $v = (v^1, v^2, \ldots v^Z)$, telle que :

$$v^i = \begin{cases} 1 & \text{si } u_1^i \neq u_2^i \\ 0 & \text{si } u_1^i = u_2^i \end{cases}, \qquad \forall i \in \{1.2, \ldots, Z\} \tag{3.9}$$

La distance de Hamming, étant le nombre de bits différents entre u_1 et u_2, est donc notée par :

$$d_H = \sum_{i=1}^{Z} v^i, \quad v^i \in \{0.1\}, \forall i \in \{1, ..., Z\} \tag{3.10}$$

Dans la Figure 3.8, un exemple numérique détaillé est donné. Comme un point de repère, la solution de référence s^r définit l'origine de la Map (voir section 3.3.3 pour le choix de s^r). Afin de trouver la position de s^e, nous avons besoin de comparer $u(s^e)$ et $u(s^r)$ en utilisant la distance de Hamming et trouver le vecteur de Hamming $v(s^e)$. Pour un bit donné dans $v(s^e)$, il prend la valeur 1 si les bits correspondants dans $u(s^e)$ et $u(s^r)$ sont différents. Par conséquent, en sommant tous les bits de $v(s^e)$, nous obtenons la distance de Hamming $d_H(s^e) = 13$.

3.3.2.3 Fonction de Mapping

La distance de Hamming d'une solution permet de décrire ses déplacements par rapport à la solution de référence s^r. Cependant, on perd alors la bijection entre l'espace de solution et la Map, car différentes solutions peuvent partager la même distance de Hamming. C'est pourquoi une *fonction de Mapping* est mise en place pour localiser les solutions sur la Map selon leurs structures.

Pour une solution s, on note $v(s) = \{v_s^1, v_s^2, \ldots, v_s^Z\}$ le vecteur de Hamming trouvé par s et s^r, avec $Z = |v(s)|$. La fonction de Mapping ("Mapping Function (MaF)") peut être appliquée selon l'Équation (3.11) :

$$\Gamma(s) = \sum_{j=0}^{d_H(s)-1} C_Z^j + \sum_{i=z_1}^{Z} \left[(1 - v_s^i) * C_{i-1}^{e(s,i)-1} \right] \tag{3.11}$$

$\Gamma(s)$ représente la localisation de s sur la Map, C est le coefficient binomiale défini par $C_n^k = \frac{n!}{k!(n-k)!}$, z_1 est la position où un bit "1" détecté pour la première fois, et $e(s, i)$ est le nombre de bits "1" cumulés depuis le premier jusqu'au $i^{\text{ième}}$ bit dans $v(s^e)$. Revenons à l'exemple de la Figure 3.8, la première occurrence de "1" se trouve au $3^{\text{ième}}$ bit, nous avons alors $z_1 = 3$. En outre, $e(s^e, i)$ est une valeur cumulée donc une fonction non-décroissante de i.

Dans les travaux de Autuori, Hnaien et Yalaoui [4], la MaF est prouvée *bijective*. Elle permet finalement de créer une bijection entre une solution et sa position sur la Map. La fonction est divisée en deux parties. La première somme dépende de $d_H(s)$ et elle reste la même pour toutes les

solutions ayant la même distance de Hamming. Ainsi, cette somme peut être considérée comme une position de départ partagée par ces solutions. Le deuxième terme, quant à lui, permet de distinguer de telles solutions, en analysant les structures des vecteurs de Hamming v – à partir de z_1 (i.e. le première bit "1"), pour chaque bit "0", une certaine valeur est ajoutée à $\Gamma(s)$, selon la position de ce bit.

Dans l'exemple de la Figure 3.8, la taille du décodage est $Z = 35$. La position basique de s^e revient à $\sum_{j=0}^{d_H(s^e)-1} C_Z^j = \sum_{j=0}^{12} C_Z^j = 1538132224$, comme toutes les autres solutions avec $d_H = 13$. Ensuite, à partir de la $3^{\text{ième}}$ position, nous faisons attention aux bits "0". Quand $i = 4$, nous avec $v_{s^e}^4$ et $e(s^e, i) = 1$ alors nous ajoutons $C_{i-1}^{e(s^e,i)-1} = C_3^0 = 1$ à $\Gamma(s^e)$. De même, lorsque $i = 5$, nous devons ajouter $C_{i-1}^{e(s^e,i)-1} = C_4^0 = 1$ à $\Gamma(s^e)$. Le bit suivant ($6^{\text{ième}}$) n'a pas de nouvelle contribution pour $\Gamma(s^e)$ car il est égale à 1, et son effet est déjà inclus dans la distance de Hamming. Au $7^{\text{ième}}$ bit, nous pouvons calculer $C_{i-1}^{e(s^e,i)-1} = C_6^1 = 6$ car $e(s^e, i) = 2$. La position de s^e est désormais incrémentée de 6. Avec le même principe, on peut trouver sa position finale $\Gamma(s^e) = 1819665515$.

3.3.2.4 Stratégies de Mapping

Avec la solution de référence s^r et la fonction de Mapping Γ, les solutions sont projetées sur la Map de façon bijective. Elles sont ensuite catégorisées en plusieurs zones – celles avec la même distance de Hamming sont regroupées dans la même zone. En outre, les rangs de Pareto des solutions sont aussi marqués sur la Map.

Dans la Figure 3.9, un exemple avec 30 solutions et 14 zones visitées est illustré. Faisons l'hypothèse que s^r trouvée par ces 30 solutions soit la même comme dans la section précédente. Dans la Figure, nous donnons également des exemples de projection pour les solutions utilisées dans la section précédente. D'une part, nous calculons les valeurs de Γ pour les solutions (abscisse) ; d'autre part, on met en évidence leurs rangs de Pareto (ordonnée). Ainsi, plusieurs types de zones peuvent être identifiées, selon Autuori et al. [4] :

- Zone non-explorée ("Unexplored Zone (UZ)") : aucune solution n'est trouvée dans la zone
- Zone explorée ("Explored Zone (EZ)") : la zone contient au moins une solution
  - Zone non-dominée ("Non-Dominated Zone (NDZ)") : la zone contient au moins une solution non-dominée
  - Zone largement explorée ("Largely Explored Zone (LEZ)") : Le nombre de solution dans la zone est supérieur au niveau moyen – le nombre moyen de solutions trouvées dans les zones explorées (qui est égal à 2.1 dans notre exemple)

Au-delà de l'identification des zones, la MaM choisit, horizontalement les zones d'intérêt pour la recherche locale, et verticalement toutes les solutions appartenant à ces zones. Pour ce faire, plusieurs stratégies du choix de solutions sont déterminées :

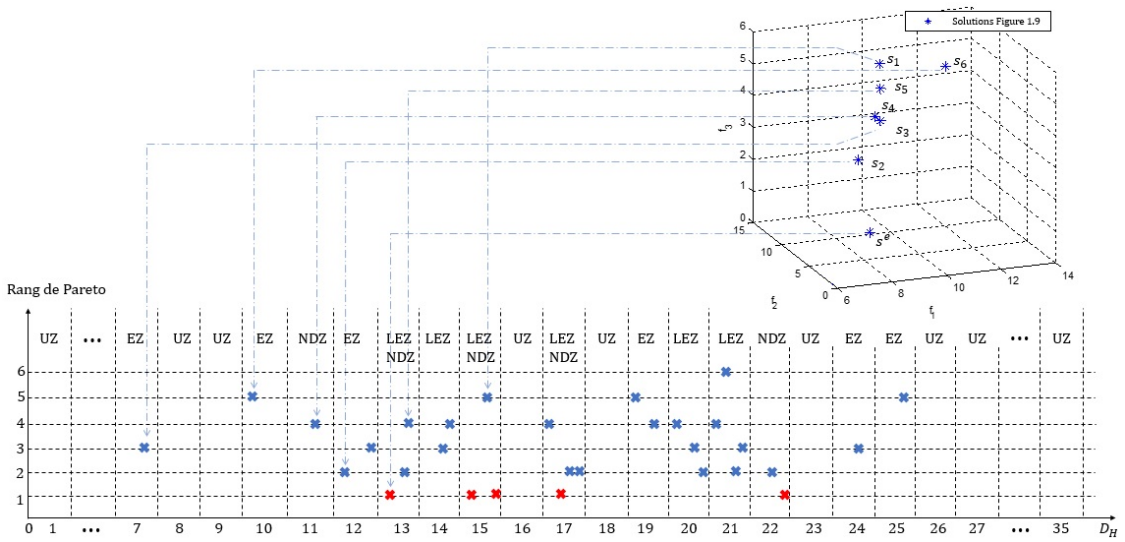

Figure 3.9 – Composer la solution de référence – un exemple

- Map1 : appliquer la recherche locale sur les solutions dans NDZ
- Map2 : appliquer la recherche locale sur les solutions dans LEZ
- Map3 : appliquer la recherche locale sur les solutions dans UZ
- Map4 : appliquer la recherche locale sur les solutions dans NDZ et UZ
- Map5 : appliquer la recherche locale sur les solutions dans LEZ et UZ

Parmi lesquelles, les Map3, Map4 et Map5 cherchent explorer des solutions qui se situent potentiellement dans les zones non-visitées (UZ). Pour ce faire, une fonction de perturbation est mise en place, comme expliqué dans l'Algorithme 3.

Cette procédure de perturbation sollicite à nouveau la méthode d'insertion. A partir de s^r, toutes les tâches sont considérées, l'une après l'autre, en tant que tâche d'insertion j^{in}. Ensuite, nous considérons chaque tâche de référence j^r valide vis à vis de j^{in}. Après l'insertion, une solution s est obtenue. Si elle appartient à une zone UZ, s est prise pour exploration du voisinage lors de la recherche locale et nous exclurons la zone correspondante de l'ensemble de UZ. De ce fait, cette procédure peut être considérée comme une mutation complète de s^r, où les nouvelles solutions situant dans UZ sont prises en compte dès trouvées.

En effet, les 5 stratégies présentées permettent de diversifier ou d'intensifier les solutions, selon les zones choisies. Plus concrètement, lorsque l'on décide d'implémenter Map1 ou Map2, on cherche à intensifier la distribution des solutions près de NDZ ou LEZ. Au contraire, si UZ sont choisies pour la recherche locale (Map3), les solutions ont une tendance à se diversifier. Et si on veut combiner l'intensification et la diversification, nous avons les 2 Maps qui couvrent NDZ+UZ (Map4) et LEZ+UZ (Map5).

Algorithm 3 Fonction de perturbation – UZ

Input : s^r – solution de référence
Output : S^{UZ} – ensemble de solutions trouvées situant dans UZ
Variables : j^{in} – tâche choisie pour effectuer l'insertion
j^r – tâche de référence
j^{dp} – premier prédécesseur achevé de j^{in}
j^{ps} – dernier successeur commencé de j^{in}

Début
$S^{UZ} = \varnothing$
Pour $a = 1, ..., n$ **faire**
  $j^{in} = V(s^r, 2a)$
  Trouver j^{dp} et j^{ps}
  Pour $b = p(j^{dp})/2 + 1, \ldots, p(j^{ps})/2 - 1$ **faire**
    $j^r = V(s^r, 2b)$
    Faire insertion avec $p(j^{in}$ et j^r (section 2.4.3.2) et obtenir s
    Si s se situe dans UZ **alors**
      $S^{UZ} = S^{UZ} \cup \{s\}$
      Mettre à jour UZ
    Fin si
  Fin pour
Fin pour
Retourner S^{UZ}
Fin

3.3.3 Amélioration de solution de référence

Dans la version de Autuori et al. [4], la solution de référence s^r est choisie aléatoirement de la population.

Dans nos études, nous proposons de créer une solution fictive en tant que référence, basée sur tous les individus de la population.

Comme explicité dans [4], le nombre théorique de solutions dans la zone z_n est $C_Z^n - C_Z^{n-1}, \forall n \in \{1, 2, \ldots, Z\}$, où Z est le nombre total de bits. Autrement dit, les zones qui sont proches et éloignées de s^r contiennent potentiellement beaucoup moins de solutions que celles qui se situe dans la région au milieu sur la Map (par exemple, dans la zone $z_{\frac{Z}{2}}$). De ce fait, il convient d'inclure les solutions intéressantes dans la régions au milieu de la Map.

Dans MaM, la distance de Hamming est liée au nombre de mouvements de déplacement par rapport à la solution de référence, et donc avec la structure des chromosomes. D'autre part, lors du déroulement de l'algorithme évolutionnaire, les solutions donnant les meilleurs résultats sont les plus probablement choisies. Il est donc intéressant de créer une solution de référence qui n'est ni trop proche ni trop loin des "bonnes" structures.

Par conséquent, pour chaque gène non-fictif dans le chromosome, nous choisissons *la valeur la moins souvent observée*. En d'autres termes, pour une position donnée dans le chromosome, nous prenons la valeur qui est présente mais le moins souvent.

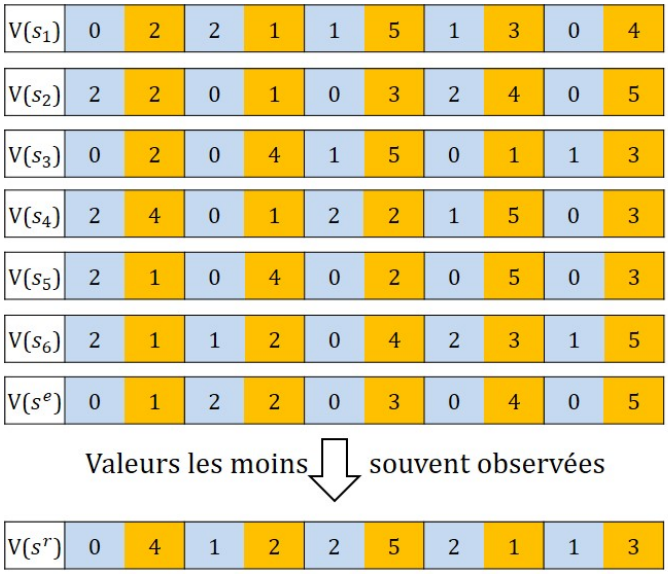

Figure 3.10 – Composer la solution de référence – un exemple

En regard de l'exemple du problème introduit dans la section 1, nous donnons 7 individus valides dans la Figure 3.10 – notés par s^e et $s_1 - s_6$ pour trouver la solution de référence s^r. Rappelons qu'un chromosome est décrit par deux types de gènes : les gènes X représentant les tâches et les gènes l pour les temps de latence, comme illustré en haut de la Figure 3.10.

Commençons par le début, pour l_1, "0" et "2" sont détectés 3 et 4 fois, respectivement. La valeur "0" est donc la valeur la moins fréquente entre les deux, c'est pourquoi nous la choisissons comme valeur de référence. Ensuite, pour X_1, premier gène X, les tâches 1 (3 fois), 2 (3 fois) et 4 (1 fois) sont observées, alors la tâche 4 est choisie. On considère par la suite l_2, où trois valeurs sont présentées : 0 (4 fois), 1 (1 fois) et 2 (2 fois). Nous mettons alors $l_2(s^r) = 1$. Regardons à présent X_2, la tâche 4 est déjà attribuée à X_1 donc elle n'est plus effective. Les tâches observées et effectives sont la tâche 1 (3 fois) et la tâche 2 (2 fois). Nous choisissons alors la tâche 2 pour X_2. De la même manière, les autres gènes de s^r peuvent être trouvés.

Pendant cette procédure, quelques cas spéciaux peuvent demander plus d'attention. Pour un gène X, seule les tâches *non-ordonnancées* sont prises en compte ; si toutes les tâches observées sont déjà séquencées dans s^r, nous prenons désormais une tâche disponible de manière aléatoire pour cette position. Par ailleurs, pour un gène donnée (X ou l), si plusieurs valeurs ont le même nombre d'occurrence, une parmi elles est choisie aléatoirement.

Par exemple, quand on considère X_3, la tâche la moins observée est la tâche 4 alors qu'elle est déjà prise pour X_1. Nous pouvons seulement choisir parmi la tâche 1 (absente), la tâche 3 (présente 2 fois) et et la tâche (détectée 2 fois) – alors la tâche 5 est choisie aléatoirement.

En prenant en compte la fréquence des valeurs observées, nous cherchons une s^r appropriée

pour une meilleure exploration de l'espace de solution. La structure de s^r est alors déterminée par tous les individus. Mais cela ne signifie pas forcément que s^r est une solution déjà existante dans la population. En ce qui concerne les gènes X, comme nous mettons l'accent sur la fréquence observée des tâches au lieu des relations de précédence parmi elles, il est toutefois possible d'obtenir une solution non réalisable. Cependant, il convient de rappeler que l'objectif de constituer s^r est d'offrir une bonne référence qui permet de positionner les solutions existantes sur le Map. C'est pourquoi la faisabilité de s^r n'est pas exigée dans cette partie.

3.3.4 Conditions d'implémentation de la recherche locale

Après avoir vu les éléments techniques, retournons à la toute première question – Quand faut-il appliquer la recherche locale ?

La procédure de recherche locale permet de sortir de l'optimum local éventuel quand l'algorithme est bloqué dans un certain espace et n'arrive à améliorer les résultats. Cela signifie que la recherche locale est plutôt intéressante quand l'évolution des solutions est faible.

C'est pourquoi nous calculons, comme dans les travaux de Autuori et al. [4], un indicateur basé sur la métrique C (voir section 1). Plus concrètement, nous calculons un indicateur d'évolution, noté par Ind^e, toutes les δ itérations comme dans l'Équation (3.12).

$$Ind^e = C(PFA_i, PFA_{i-\delta}) \tag{3.12}$$

où PFA_i est le front de Pareto actuel à l'itération i. Ensuite, un seuil Ind^{min} est choisi – si $Ind^e < Ind^{min}$, on procède à la recherche locale ; sinon, elle n'est pas implémentée car l'évolution des solutions est encore suffisamment active.

3.4 Tests numériques et conclusion

Pour le problème multi-objectif, nous nous concentrons sur la résolution approchée. Le NSGAII et le NSGAIII sont d'abord mis en comparaison pour étudier les effets des différents "niching" mécanisme. Ensuite, une recherche locale est intégrée dans chacune des méthodes pour améliorer les fronts approchés précédemment trouvés.

3.4.1 Les approches NSGA

3.4.1.1 Paramétrage

Avant de lancer les tests, nous envisageons une calibration pour les paramètres des algorithmes génétiques. Plus concrètement, 5 facteurs sont à analyser – la taille de la population N, le nombre de générations n_G, le taux de croisement c_r, le taux de mutation m_r et le nombre de segments n_S pour affiner le plan de référence (seulement impliqué dans NSGAIII).

Tableau 3.1 – Paramétrage

	N	n_G	c_r	m_r	n_S	(N, n_G, c_r, m_r, n_S)
$G30$	100	200, 400, 600	0.85	0.05	10	(200, 600, 0.95, 0.10, 20)
$G60$	150	400, 600, 800	0.90	0.10	15	(200, 600, 0.95, 0.15, 15)
$G90$	200	600, 800, 1000	0.95	0.15	20	(200, 1000, 0.85, 0.15, 10)
$G120$		600, 800, 1000				(200, 1000, 0.90, 0.10, 10)

Nous testons 3 valeurs possibles pour chaque paramètres, cela entraîne un nombre de combinaisons possibles très élevé (3^5). Nous appliquons donc la méthode de Taguchi [70] pour simplifier le plan d'expériences. Dans le tableau 3.1, nous mettons en évidence les valeurs testés pour chaque facteur et aussi les combinaisons choisies. Pour les petites instances, le critère principal est la qualité de solution ; pour moyennes instances, nous considérons le compromis entre la qualité de solution et le temps de calcul ; en ce qui concerne les grandes instances, le temps de calcul devient l'aspect principal lors de la calibration.

Le NSGAIII est choisi comme la méthode de référence comme il contient tous les facteurs considérés. Le tableau $L27(9*3)$ de Taguchi [70] est utilisé, avec 15 instances aléatoirement choisies lors de la calibration pour chaque groupe. Nous prenons en compte la "moyenne" comme premier critère mesurant la qualité des solutions. Le rapport "signal/bruit (S/B)" n'est pas en contradiction avec ce premier dans le plupart des cas testés. En cas des performances quasi-équivalentes, on considère le rapport comme critère secondaire.

Comme exemple, nous montrons dans les Figures 3.11 – 3.14 les détails pour la calibration du $G90$. D'après les graphiques des effets principaux (Figure 3.11 et 3.13), la configuration choisie $N(3), n_G(3), c_r(1), m_r(3), n_S(1)$. Ensuite, comme on peut le voir, N a une influence beaucoup plus importante que les autres facteurs sur les résultats. Pour cette raison, on trace les graphiques des interactions entre N et tous les autres facteurs. Dans l'exemple donné, les interactions confirment le choix fait avec les effets principaux. Lors de la calibration du paramétrage, nous remarquons que les résultats vis à vis du rapport S/B sont dans la plupart des cas similaires avec ceux trouvés en regardant la moyenne. Ceci permet de re-confirmer les paramètres choisis.

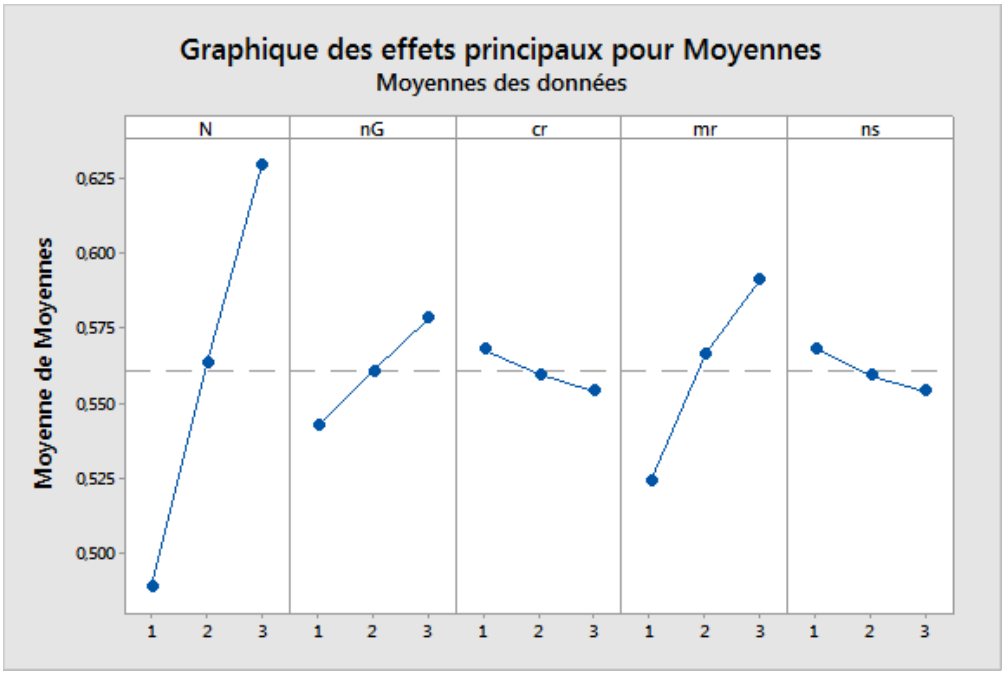

Figure 3.11 – Moyenne – effets principaux – $G90$

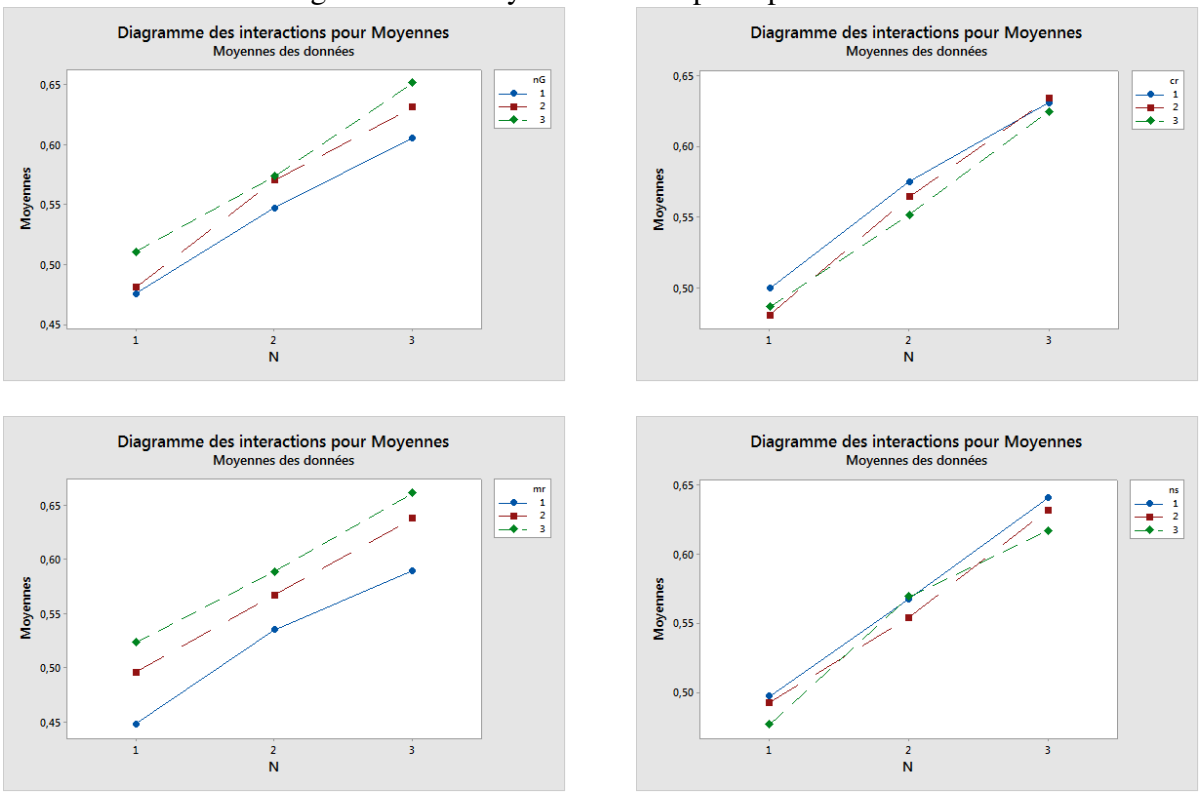

Figure 3.12 – Moyenne – interactions – $G90$

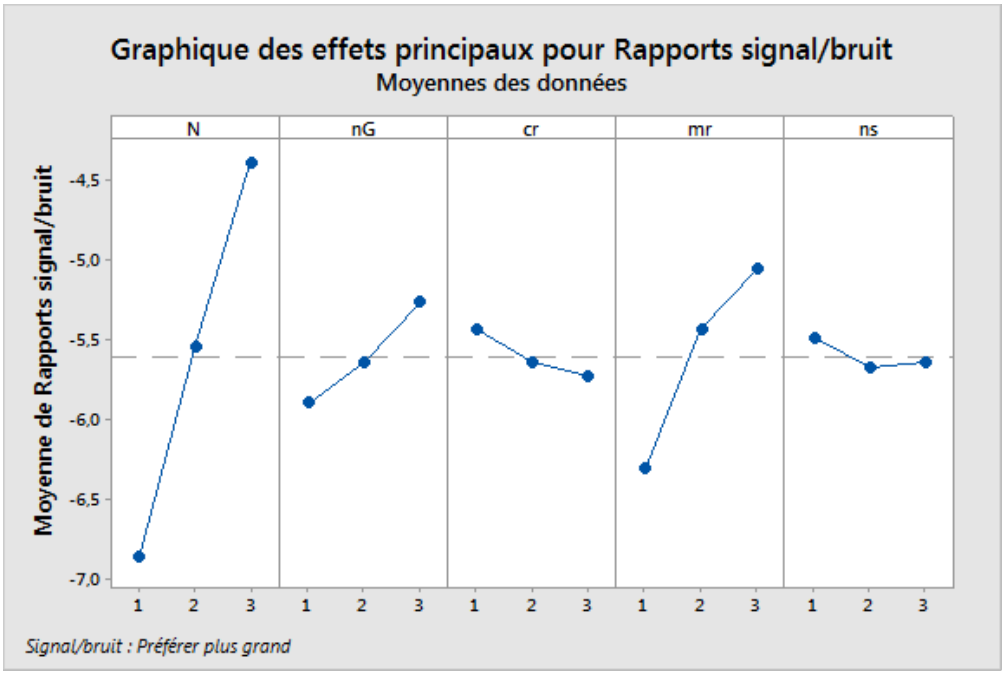

Figure 3.13 – Signal/bruit – effets principaux – $G90$

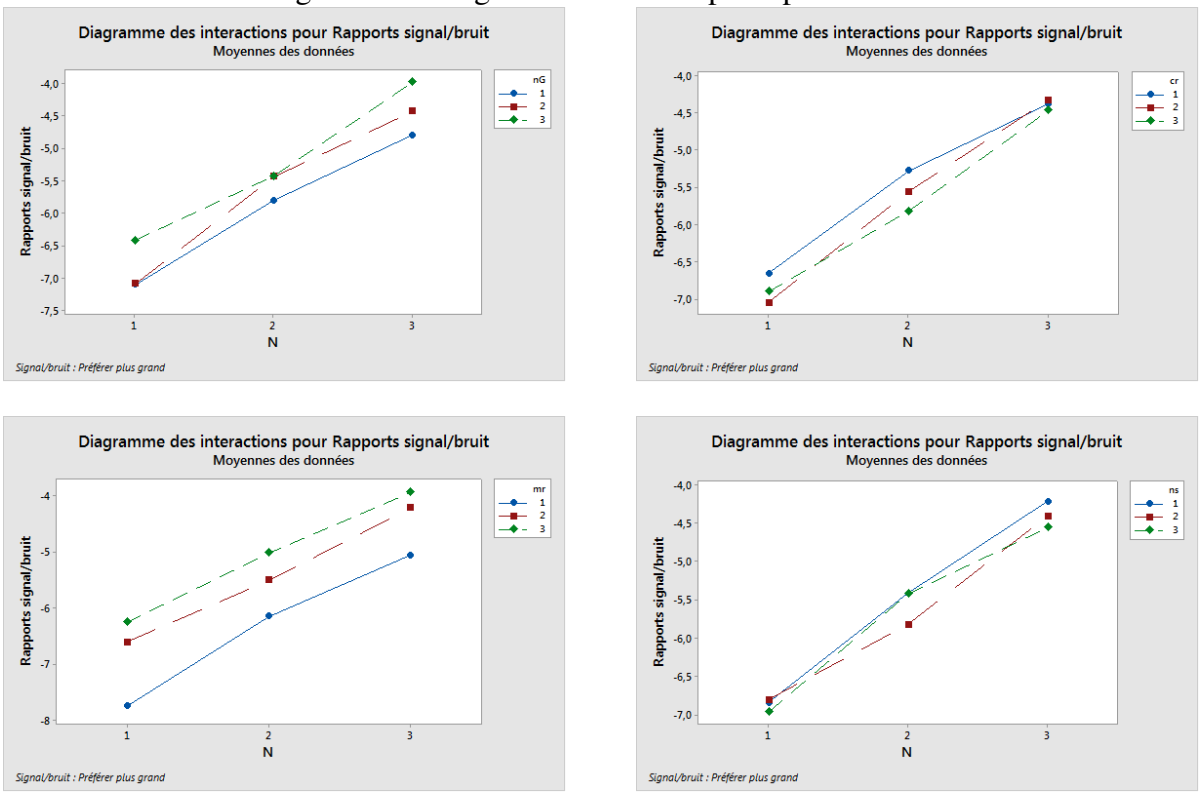

Figure 3.14 – Signal/bruit – interactions – $G90$

3.4.1.2 Résultats

Nous comparons dans un premier temps les performances de NSGAII et NSGAIII sur le problème multi-objectif. Les expériences portent sur les instances avec de 30, 60, 90, et 120 tâches, notés respectivement par G30, G60, G90, G120. Les données sont réparties et générées comme expliqué dans la section 2.5.

Les résultats sont d'abord mesurés par l'hypervolume (aussi noté par hv) afin d'évaluer les fronts de Pareto approchés selon leurs régions de dominance dans l'espace de recherche (voir section 1.4). La solution Nadir est utilisée comme le point de référence partagé par les deux méthodes. Rappelons brièvement que $hv \in (0, 1)$, où 1 représente la couverture complète et 0 signifie que le front ne domine rien (d'autre que le point Nadir).

Dans le Tableau 3.2, nous comparons le NSGAII et le NSGAIII, notés respectivement par M1 et M2 dans la suite du rapport. Dans le tableau, nous donnons les moyennes du hypervolume de chaque sous-groupe, trouvé par M1 et M2 pour G30, G60, G90 et G120. A chaque comparaison, on met en gras la méthode qui est meilleure parmi les deux. En bas du tableau, nous marquons la moyenne globale (ligne "Moyenne"), le nombre de sous-groupes où chaque méthode trouve meilleur front (ligne "Meilleur"), et le nombre de sous-groupes où les deux méthodes finissent par le même hypervolume.

De même, dans la Figure 3.15, nous mettons en évidence l'évolution la moyenne de l'hypervolume des méthodes en fonction des paramètres α, β et γ. Pour chaque sous-groupe $\alpha\beta\gamma$, on calcule la moyenne de tous les groupes (G30, G60, G90 et G120). Comme on peut le voir sur la Figure 3.15, le NSGAIII est plus avantageux dans la plupart des cas. De plus, les groupes où le NSGAII est meilleur se trouvent surtout aux instances relativement faciles à résoudre ($\alpha = 1$). Autrement dit, le NSGAIII a une meilleure performance globale que le NSGAII.

Retournons sur le Tableau 3.2, les détails sur chaque groupe et sous-groupe de tests confirment nos observations précédentes vis-à-vis de la performance globale. D'abord, M2 propose une meilleure moyenne pour chaque groupe de tests que M1. Quand nous détaillons les résultats des sous groupes, nous remarquons que M1 trouve un meilleur front approché que M2 dans la plupart des cas. Cet avantage est surtout évident pour G60 et G120, où les nombres respectifs de sous-groupes mieux résolus par M2 sont 16 et 19 parmi le total de 27. De même, M2 apporte une amélioration de 6% sur la valeur moyenne globale de M1. Sur G30 et G90, les deux méthodes proposent des performances plutôt similaires. Toutefois, M2 reste la meilleure parmi les deux, en tenant compte de la valeur moyenne et surtout le nombre de sous-groupes mieux résolus. De plus, lorsque la taille du problème augmente, on trouve moins de sous-groupes avec les hypervolumes identiques des deux méthodes. Cela signifie que sur les problèmes de petites tailles, M1 et M2 se

Tableau 3.2 – Comparaison M1 et M2, hypervolume

hv	G30		G60		G90		G120	
$\alpha\beta\gamma$	M1	M2	M1	M2	M1	M2	M1	M2
111	0.57	**0.60**	0.38	**0.51**	0.44	**0.47**	**0.62**	0.45
112	0.54	**0.56**	0.42	**0.54**	**0.44**	0.36	**0.50**	0.45
113	**0.66**	0.65	0.50	**0.51**	**0.56**	0.41	**0.53**	0.39
121	**0.60**	**0.60**	**0.48**	0.42	**0.42**	0.41	0.50	**0.54**
122	**0.62**	0.61	0.47	**0.54**	0.39	**0.51**	**0.47**	0.44
123	0.56	**0.59**	0.41	**0.50**	0.33	**0.47**	**0.42**	0.41
131	0.57	**0.62**	0.44	**0.54**	0.48	**0.53**	0.45	**0.54**
132	**0.59**	**0.59**	**0.53**	0.45	**0.47**	0.41	**0.62**	0.54
133	0.51	**0.54**	**0.51**	0.45	**0.51**	0.50	0.41	**0.44**
211	0.45	**0.48**	**0.54**	0.50	**0.45**	0.39	0.38	**0.44**
212	0.47	**0.50**	0.48	**0.54**	**0.39**	0.36	0.44	**0.47**
213	0.46	**0.49**	**0.51**	0.47	0.36	**0.39**	**0.44**	0.47
221	0.51	**0.53**	0.39	**0.48**	**0.53**	0.50	0.42	**0.45**
222	**0.48**	0.47	**0.41**	**0.41**	**0.47**	0.35	0.50	**0.56**
223	0.53	**0.54**	0.41	**0.45**	0.36	**0.56**	0.38	**0.39**
231	0.49	**0.53**	0.50	**0.54**	0.56	**0.62**	0.39	**0.56**
232	**0.57**	0.53	0.50	**0.51**	**0.54**	0.51	0.39	**0.50**
233	**0.54**	**0.54**	0.57	0.54	0.59	**0.60**	0.50	**0.53**
311	**0.54**	0.53	**0.45**	**0.45**	0.41	**0.47**	0.39	**0.42**
312	**0.53**	0.48	**0.54**	0.48	0.35	**0.54**	0.38	**0.48**
313	**0.62**	0.51	0.39	**0.48**	**0.56**	0.44	0.36	**0.47**
321	**0.56**	0.44	**0.39**	0.38	0.38	**0.42**	0.47	**0.54**
322	0.59	**0.60**	0.38	**0.47**	0.41	**0.45**	0.42	**0.57**
323	**0.56**	0.50	0.41	**0.42**	**0.36**	0.35	0.38	**0.45**
331	**0.68**	0.63	0.54	**0.69**	0.56	**0.68**	**0.65**	0.62
332	0.48	**0.51**	**0.65**	0.59	**0.68**	0.63	**0.77**	0.71
333	0.51	**0.62**	0.51	**0.60**	0.41	**0.56**	0.52	**0.69**
Moyenne	0.55	0.55	0.47	0.50	0.46	0.47	0.47	0.50
Meilleur	10/27	14/27	9/27	16/27	13/27	14/27	8/27	19/27
Equivalent	3		2		0		0	

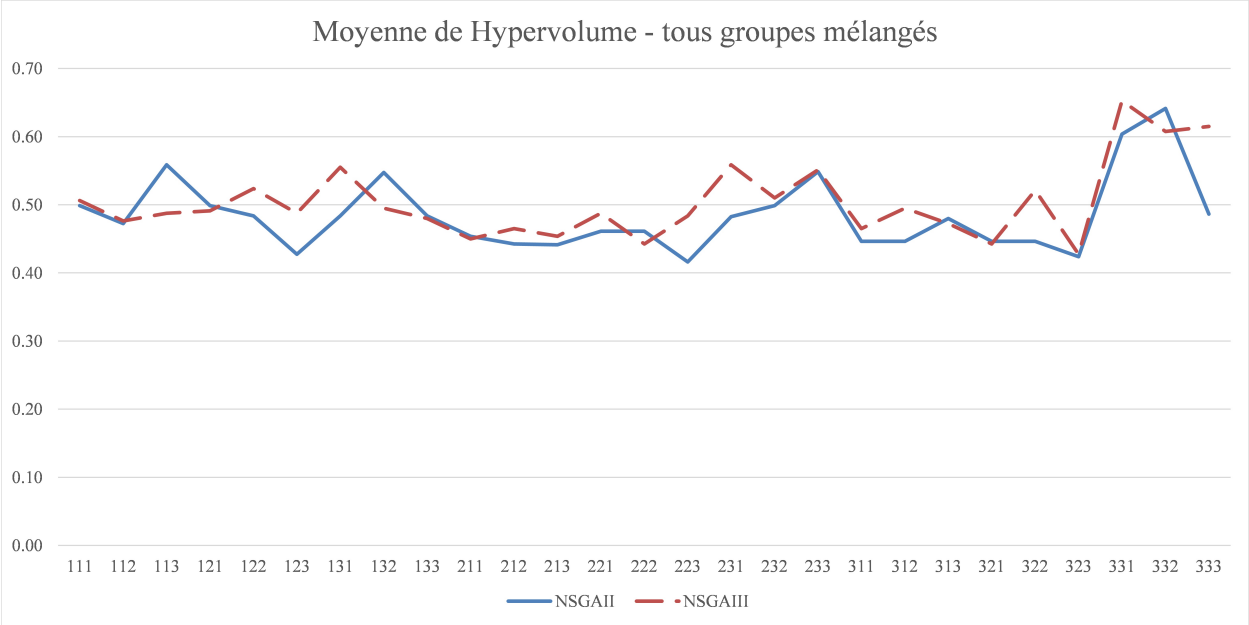

Figure 3.15 – Comparaison M1 et M2, hypervolume

rejoignent sur le même front approché. Toutefois, quand le nombre de tâches devient élevé, les algorithmes possèdent des performances différentes de plus en plus.

Par conséquent, comme remarqué dans la section 2.5.3, il est nécessaire de mettre en place des techniques d'amélioration pour mieux explorer l'espace de recherche.

3.4.2 Recherche locale

Dans cette section, nous mettons en œuvre la recherche locale avec le technique de Mapping. Récemment proposé par Autuori, Hnaien et Yalaoui (2016), son efficacité a été soulignée sur un problème Job-Shop Flexible [4]. Nous adaptons dans un premier temps cette méthode sur notre problème dans la section 3.4.2.1 et comparons les 5 stratégies d'exploration (voir section 3.3.2.4). Ensuite, la section 3.4.2.2 est consacrée à une comparaison entre le point de référence proposé dans ce travail (voir section 3.3.3) et celui utilisé dans les travaux de Autuori et al. [4] afin de déterminer l'efficacité de nouvelle solution de référence.

3.4.2.1 Comparaison des stratégies de Mapping

Mettons en œuvre maintenant la recherche locale avec la MaM. Nous considérons les 5 stratégies de Mapping comme présentés dans la section 3.3.2.4 et on les notes comme montré dans le Tableau 3.3.

Tableau 3.3 – Notations des méthodes de Mapping

Sans MaM	Map1	Map2	Map3	Map4	Map5
M1	M1-1	M1-2	M1-3	M1-4	M1-5
M1	M2-1	M2-2	M2-3	M2-4	M2-5

Nous utilisons à nouveau l'hypervolume pour évaluer leurs performances et présentons les résultats dans le Tableau 3.4. Dans le tableau, au lieu de détailler tous les sous-groupes comme dans la section précédente, nous synthétisons dans chaque case, la valeur moyenne de toutes les instances du groupe correspondant. La moyenne globale vis-à-vis de tous les groupes de tests est calculée pour chaque méthode, et marquée dans la ligne "Moyenne". Ces valeurs nous permettent finalement d'attribuer un classement pour toutes les méthodes testées.

Tableau 3.4 – Hypervolume comparaison

Groupe	M1	M1-1	M1-2	M1-3	M1-4	M1-5	M2	M2-1	M2-2	M2-3	M2-4	M2-5
G30	0.51	0.58	0.60	0.52	0.59	**0.62**	0.51	0.58	0.60	0.52	0.58	**0.62**
G60	0.43	0.51	0.58	0.45	0.50	0.58	0.47	0.51	**0.59**	0.45	0.51	**0.59**
G90	0.40	0.46	0.51	0.40	0.45	**0.53**	0.41	0.44	**0.53**	0.42	0.44	**0.53**
G120	0.35	0.40	0.47	0.38	0.40	**0.48**	0.39	0.40	**0.48**	0.39	0.42	0.45
Moyenne	0.42	0.49	0.54	0.44	0.48	**0.55**	0.44	0.48	**0.55**	0.45	0.49	**0.55**
Rang	12	5	4	10	7	**1**	10	7	**1**	9	5	**1**

De manière générale, la MaM permet d'améliorer le schéma original des algorithmes NSGAII et NSGAIII, où l'efficacité des Maps se différencie. En mettant l'accent sur la meilleure méthode de chaque groupe, M1-5, M2-2 et M2-5 sont classées au 1^{er} rang avec le même hypervolume en moyenne. Chacune de ces trois méthodes permet aussi de trouver les meilleurs résultats dans 3 groupes de tests parmi le total de 4. Par rapport au schéma original de l'NSGA, elles proposent une amélioration de 31% pour M1 et de 25% pour M2. Légèrement moins performante, M1-2 améliore M1 et M2 de 28.6% et 22.7%. Parmi les 5 stratégies de Mapping, le Map5 et le Map2 sont meilleures que les autres. M3 est la moins efficaces mais proposent tout de même des améliorations.

Comme un complément du hypervolume, on utilise le métrique C pour comparer les méthodes par paire. Par exemple, pour deux méthodes MA et MB, la métrique C permet d'évaluer parmi deux fronts non-dominés F_{MA} et F_{MB} à l'aide de la dominance de Pareto, tel que :

$$C(MA, MB) = \frac{|s_a \in F_{MA} : \exists s_b \in F_{MB}, s_b \prec_P s_a|}{|F_{MA}|}$$

En d'autres termes, elle calcule pour F_{MA}, la proportion des solutions dominées par F_{MB}.

Comme expliqué dans la section 1.4, calculons dans nos études le ratio des métriques C pour expliciter la performance des deux méthodes concernées :

$$C_r(MA, MB) = \frac{C(MA, MB)}{C(MB, MA)}$$

Par conséquent, $C_r(MA, MB) < 1$ implique que MA est meilleure parmi les deux ; dans le cas contraire, c'est MB qui serait plus performante.

Dans le Tableau 3.5, nous présentons les résultats de C_r pour tous les couples parmi les 12 méthodes. Pour une case $C_r(MA, MB)$, MA est la méthode qui se trouve dans la ligne correspondante et MB est celle de la colonne. Par exemple, nous avons $C_r(M1-1, M1) = 0.43$ et on peut en déduire que M1-1 donne de meilleurs fronts approchés que M1.

En effet, le nombre de comparaisons dans le Tableau 3.5 est assez élevé vu le nombre de méthodes prises en compte (12 méthodes donc 132 comparaisons dans le tableau). Pour mettre en évidence les informations critiques, nous proposons la Figure 3.16 comme une version "allégée".

Les méthodes sont classées selon leur famille, on les compare d'abord à l'intérieur de chacune des familles. Pour chaque couple comparé, la flèche pointe vers celle qui est meilleure. En même temps, on marque un C_r des deux méthodes correspondantes, de celle en amont à celle en aval. Par exemple, quand on considère M1 et M1-2, la flèche pointe vers M1-2, qui est beaucoup plus performante que M1. La valeur du C_r explicitée est 2.52, qui peut être considérée comme un poids d'amélioration. Plus le poids est grand, plus la méthode en aval améliore celle en amont. De la même façon, on extrait les combinaisons intéressantes et détermine la meilleure méthode de chaque famille – M1-2 et M2-5. Ensuite, nous comparons ces deux méthodes et obtient la méthode la plus efficace parmi tous – M2-5, qui améliore M1-2 avec un poids de 1.09.

Toutefois, nous avons $C_r(M1-5, M1-2) = 1.01$, qui signifie que leur performances restent très proche. On remarque également que dans chaque famille, Map2 et Map5 sont plus efficaces que Map1 et Map4. Map3 est le moins performant, pourtant il permet tout de même d'améliorer les méthodes de base. Ceci signifie que la plus importante contribution est apportée par LEZ (partie en commun de Map2 et Map5), que UZ permet de renforcer la performance au-delà de LEZ.

Revenons à présent sur le Tableau 3.5. En premier lieu, on met l'accent sur les meilleures améliorations pour M1 et M2 par des lettres en gras. Il s'agit de M2-5 et cette hybridation montre à nouveau la meilleure performance parmi toutes les méthodes testées. Les trois autres méthodes intéressantes selon la Figure 3.16 sont M1-2 et M1-5 et M2-2. On note leurs performances en lettres *italiques* dans le Tableau. Comme on peut le voir, elles ne sont pas aussi puissantes que M2-5, mais restent beaucoup plus efficaces que les autres méthodes.

En résumé, nous pouvons conclure les que 4 meilleures méthodes dans l'ordre sont M2-5, M1-2, M1-5, M2-2. Ceci rejoint et complète nos remarques avec la Figure 3.16. La meilleure

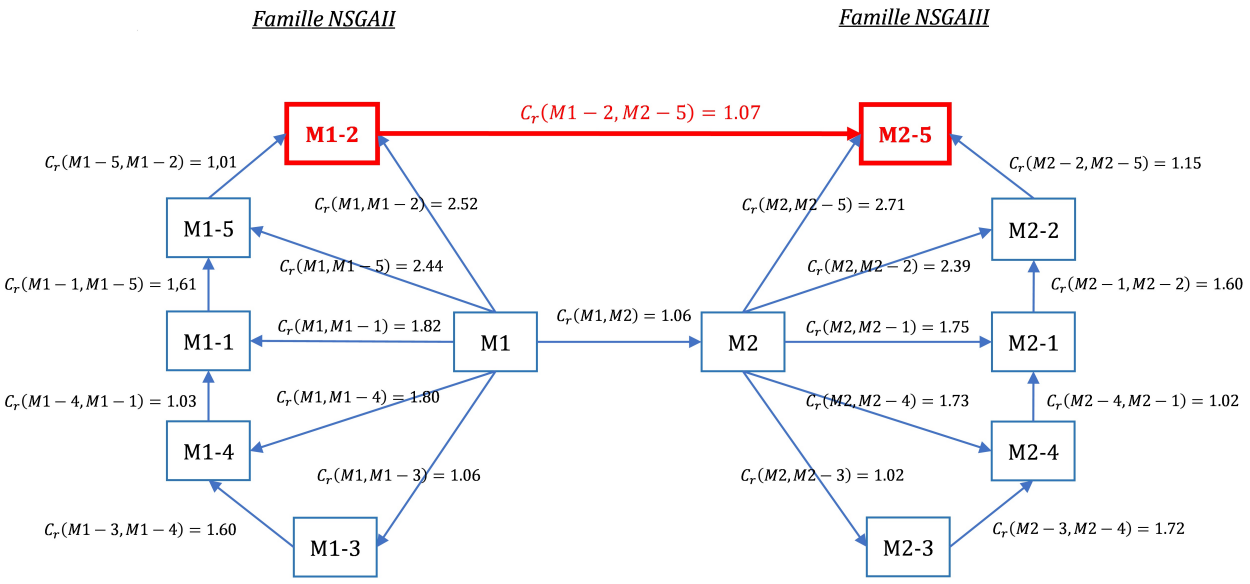

Figure 3.16 – Comparaison de la métrique C - graphique

hybridation parmi toutes méthodes est M2-5. Le Map5 et le Map2 sont beaucoup plus efficaces que les autres Map, mais le Map5 reste le plus puissant.

Tableau 3.5 – Comparaison de la métrique C - tous les résultats

C_r	M1	M1-1	M1-2	M1-3	M1-4	M1-5	M2	M2-1	M2-2	M2-3	M2-4	M2-5
M1	-	1.82	*2.52*	1.06	1.80	*2.44*	1.06	1.81	*2.42*	1.03	1.75	**2.70**
M1-1	0.56	-	1.62	0.65	0.98	1.61	0.61	0.97	1.51	0.63	0.97	1.70
M1-2	*0.40*	0.63	-	0.41	0.59	0.99	*0.40*	0.59	0.90	0.39	0.61	1.08
M1-3	0.96	1.57	2.59	-	1.60	2.42	0.97	1.65	2.33	0.97	1.61	2.59
M1-4	0.56	1.03	1.54	0.62	-	1.64	0.61	1.01	1.52	0.64	0.98	1.75
M1-5	*0.41*	0.61	1.01	0.44	0.61	-	*0.39*	0.66	0.94	0.40	0.63	1.05
M2	0.95	1.65	*2.74*	1.04	1.64	*2.60*	-	1.75	*2.39*	1.02	1.73	**2.72**
M2-1	0.56	1.04	1.55	0.63	1.00	1.54	0.58	-	1.60	0.62	0.99	1.75
M2-2	*0.41*	0.67	1.08	0.43	0.66	1.07	*0.42*	0.63	-	0.41	0.72	1.15
M2-3	0.98	1.61	2.73	1.05	1.66	2.50	0.99	1.62	2.57	-	1.72	2.76
M2-4	0.59	1.03	1.53	0.62	1.02	1.62	0.59	1.02	1.41	0.59	-	1.77
M2-5	**0.38**	0.59	0.95	0.40	0.57	0.96	**0.37**	0.57	0.87	0.37	0.56	-

3.4.2.2 Amélioration de solution de référence

Le schéma de Mapping proposé par Autuori, Hnaien et Yalaoui [4] est efficace sur notre problème. Par la suite, nous testons une nouvelle solution de référence proposée dans ce travail sur le Map5, et on cherche éventuellement à améliorer encore les résultats.

Rappelons que dans l'étude de Autuori et al. [4], la solution de référence du Mapping est choisi

Tableau 3.6 – Reference solutions – hypervolume

	M1-5$_{ref}$	M2-5$_{ref}$	M1-5	M2-5
G30	0.57	0.55	0.51	0.52
G60	0.60	0.56	0.48	0.47
G90	0.50	0.52	0.42	0.41
G120	0.43	0.48	0.40	0.41
Av.	**0.52**	**0.53**	**0.45**	**0.45**

Tableau 3.7 – Métrique C, comparaison des solutions de référence

c_r	M1-5$_{ref}$	M2-5$_{ref}$	M1-5	M2-5
M1-5$_{ref}$	-	1.32	**0.80**	0.86
M2-5$_{ref}$	0.77	-	0.52	**0.70**
M1-5	**1.26**	1.98	-	1.13
M2-5	1.17	**1.45**	0.88	-

aléatoirement dans la population. Dans ce travail, nous créons une solution de référence à partir des solutions, comme expliqué dans la section 3.3.3. Nous intégrons la solution de référence proposée dans la plus efficace Map – le Map5 et nous notons ces méthodes par M1-5$_{ref}$ et M2-5$_{ref}$.

Le Tableau 3.6 compare M1-5$_{ref}$ et M2-5$_{ref}$ avec M1-5 et M2-5 selon l'hypervolume. Pour chaque groupe de tests, M-1$_{ref}$ (M-2$_{ref}$) réussit à trouver meilleurs fronts de Pareto que M1-5 (M2-5). En moyenne, la solution de référence proposée dans ce travail améliore celle choisie de façon aléatoire par respectivement 22% et 24%, vis-à-vis du schéma NSGAII et NSGAIII.

Ensuite, nous présentons la métrique C dans le Tableau 3.7. On met l'accent sur les comparaisons entre (M1-5, M1-5$_{ref}$) et (M2-5, M2-5$_{ref}$). La métrique confirme les résultats de l'hypervolume. Globalement, la solution de référence proposée apporte une amélioration significative. D'ailleurs, on note que cette amélioration est plus efficace sur le schéma NSGAIII que NSGAII.

Suite aux améliorations confirmées, toutes les expériences avec la MaM par la suite intègrent la solution de référence proposée.

3.4.2.3 Conclusion

Dans ce chapitre, nous avons introduit un nouvel objectif, l'équilibrage d'allocation des ressources, et proposé un problème multi-objectif correspondant. Nous avons d'abord complété la formulation mathématique utilisée dans le chapitre précédent. Ensuite, le NSGAII est adapté pour résoudre le problème multi-objectif. De plus, on a aussi implémenté le NSGAIII comme une

deuxième méthode approchée. Les deux approches NSGA sont comparées selon l'hypervolume. Les tests montrent que le NSGAIII propose de meilleurs résultats que le NSGAII. En même temps, la non-similarité des fronts de Pareto trouvés par ces deux méthodes montre qu'il est intéressant de renforcer l'exploration de l'espace de recherche.

Pour ce faire, nous avons mis en place une recherche locale avec la méthode de Mapping. On s'est inspiré des travaux de Autuori, Hnaien et Yalaoui [4] et on a hybridé 5 stratégies de Mapping avec le NSGAII ainsi que le NSGAIII. Les méthodes sont d'abord comparées selon l'hypervolume. Parmi les 5 stratégies de Mapping, le Map2 (exploration dans LEZ) et le Map5 (exploration dans LEZ et UZ) sont beaucoup plus performants que les autres. L'amélioration apportée par la MaM s'élève à 31% pour le NSGAII et 25% pour le NSGAIII. D'ailleurs, on réalise des comparaisons par paire et calcule la métrique C pour compléter l'évaluation par l'hypervolume. Les observations précédentes sont confirmées et on conclut que, vis-à-vis des deux métriques, l'hybridation entre le NSGAIII et le Map5 domine toutes les autres méthodes. En effet, le Map5 permet d'une part d'intensifier l'exploration dans les zones les plus visitées, d'autre part, il cherche la diversification avec une recherche des solutions situant dans les zones non visitées.

Par la suite, nous avons proposée une amélioration pour la MaM utilisée par Autuori, Hnaien et Yalaoui. Comme l'origine du Map, la solution de référence joue un rôle très important sur l'efficacité de la MaM. Nous avons mis en œuvre un procédure pour créer une solution de référence à la base de toute la population. Le Map5 est choisi comme représentant des stratégies de Mapping. Les tests montrent une amélioration de 22% pour le schéma du NSGAII et de 24% pour celui du NSGAIII sur l'hypervolume. Les résultats de la métrique C confirme ensuite les améliorations apportées.

A travers de tous tests, nous avons aussi remarqué que la famille du NSGAIII est plus performant que celle du NSGAII sur notre méthodes. D'ailleurs, la MaM propose une meilleure coopération avec le NSGAIII que le NSGAII.

En conclusion, dans ce chapitre, la meilleure méthode hybridée est celle entre le NSGAIII, le Map5 et la solution de référence proposée dans ce travail (noté par Map5$_{ref}$ dans la suite de ce document).

Chapitre 4

Amélioration du problème multi-objectif avec diverses règles de dominance

Dans les chapitres précédents, un problème bi-objectif ainsi qu'un problème multi-objectif ont été résolus. Les fronts de Pareto approchés de bonne qualité sont déterminés. Sans pouvoir résoudre les problèmes de manière exacte pour les instances de grandes tailles, la performance des méthodes peut être évaluée par la diversification et l'intensification des fronts obtenus.

Les fronts non-dominés trouvés avec le NSGAII et le NSGAIII proposent un ensemble de solutions très diversifiées et étirées. Cependant, pour les décideurs (*decision-makers*), il est parfois troublant d'envisager beaucoup de solutions équivalentes. Pour cette raison, nous mettons en œuvre dans nos travaux, certaines techniques afin de pré-choisir les solutions potentiellement plus intéressantes pour les DMs.

Nous considérons, alors plusieurs règles de dominance autre que la dominance de Pareto, connue en tant que la dominance conventionnelle (section 4.1). La mise en place de ces règles permettra d'affiner les fronts de Pareto trouvés par la dominance conventionnelle, et d'intégrer les préférences du choix des individus lors de l'évolution.

4.1 Règles de dominance alternatives

4.1.1 sL-dominance

La dominance de Lorenz (notée par "L-dominance") est initialement introduite par Kostreva and Ogryczak (1999) [42], et ensuite améliorée par Kostreva et al. (2004) [39]. La dominance de Lorenz prend en compte le fait que tous les objectifs ne jouent pas toujours le même rôle, et qu'un

écart de la même valeur sur des objectifs différents n'ont pas forcément la même importance pour les DMs. De manière concrète, elle propose d'optimiser les différents critères au même niveau. Par conséquent, elle est aussi appelée "la dominance équitable".

La dominance de Lorenz est récemment incluse dans les études de Dugardin, Yalaoui et Amodeo (2010) [23] pour résoudre un problème bi-objectif de type flowshop ré-entrant. Cette hybridation entre la L-dominance et le NSGAII est prouvée plus performante que le NSGAII et SPEA2. Moghaddam, Yalaoui et Amodeo (2012) [49] (2015) [47] ont aussi considéré la dominance de Lorenz comme une amélioration de la dominance de Pareto, lors de la résolution de problème d'ordonnancement ré-entrant avec deux critères, la L-dominance est à nouveau couplée avec le NSGAII.

Afin de définir la dominance de Lorenz, nous avons besoin de calculer le vecteur de Lorenz. Étant donné que les amplitudes des valeurs objectives sont potentiellement très différentes dans notre problème, nous mettons en place une version normalisée de la L-dominance (aussi dit "scaled-Lorenz Dominance", notée par "sL-dominance" ou "sLorenz dominance" par la suite). Pour ce faire, étant donné une solution s, ses valeurs objectives sont d'abord mises à l'échelle comme dans l'équation (4.1) :

$$f'_k(s) = \frac{f_k(s) - f_k^{\min}}{f_k^{\max} - f_k^{\min}}\,, \qquad \forall k \in \{1,2,3\} \tag{4.1}$$

où les valeurs objectives mises à l'échelle notées f'_k pour le critère $k \in \{1,2,3\}$, et $f_k^{\max}$ ($f_k^{\min}$) la valeur maximale (minimale, respectivement) pour $k \in \{1,2,3\}$. Comme remarque, cette étape n'a pas été prise en compte dans les travaux selon la littérature actuelle. La sL-dominance s'approche de la L-dominance lorsque les amplitudes des objectives sont proches ; et les deux reviennent au même quand les amplitudes sont exactement pareilles.

Ensuite, les f'_k sont triées par ordre décroissant comme ci dessous, telles que :

$$f'_{[1]}(s) \geq f'_{[2]}(s) \geq f'_{[3]}(s) \tag{4.2}$$

Finalement, nous pouvons écrire pour la solution s, le vecteur de sLorenz $(f_1^{sL}, f_2^{sL}, f_3^{sL})$, où le $k^{\text{ième}}$ élément est la somme des k premières valeurs de $f'_{[k]}$:

$$f_k^{sL}(s) = \sum_{i=1}^{k} f'_{[k]}(s)\,, \qquad \forall k \in \{1,2,3\} \tag{4.3}$$

L'équation (4.1) permet de gérer les différents registres des valeurs objectives et de les normaliser selon les zones d'intérêts. En même temps, le point d'origine peut être définit par $O^{sL} = (f_1^{\min}, f_2^{\min}, f_3^{\min})$. Les objectifs modifiés sont ensuite triés en fonction de leurs contributions, leurs importances selon l'équation (4.2). Enfin, le vecteur de sLorenz est calculé en cumulant les contributions des objectifs, comme dans l'équation (4.3).

Dans un problème de minimisation avec K objectifs, pour deux solutions s_1 et s_2, la dominance de sLorenz est établie si leurs vecteurs de Lorenz ne sont pas Pareto équivalent :

$$s_1 \prec_{sL} s_2 \Leftrightarrow \begin{cases} f_k^{sL}(s_1) \leq f_k^{sL}(s_2), \quad \forall k \in \{1, 2, \ldots, K\} & (4.4) \\ \exists k' \in \{1, 2, ..., K\}, \quad f_{k'}^{sL}(s_1) < f_{k'}^{sL}(s_2) & (4.5) \end{cases}$$

La sL-dominance cherche à optimiser tous les objectifs de manière équivalente, en balançant les valeurs selon leurs contributions sur les critères correspondants. En conséquence, les solutions proposant des valeurs excellentes sur certains objectifs (ce qui ramène aux mauvaises performances sur les autres objectifs) sont éliminées, voir exemple numérique ci-dessous.

Rappelons l'exemple numérique au début du chapitre, nous donnons les calculs pour déterminer les fronts de Lorenz dans le tableau 4.1. En complément, sur la figure 4.1, nous montrons le changement de Pareto à sLorenz. Les rangs de solutions selon les dominances de Pareto ou de sLorenz sont mis en évidence, dans les graphes correspondantes (notés dans les []). Nous mettons en avant les différents fronts trouvés par ces règles de dominance.

Tableau 4.1 – Numerical example – sLorenz dominance

	f_1	f_2		f'_1	f'_2		$f'_{[1]}$	$f'_{[2]}$		f_1^{sL}	f_2^{sL}	sL-rang
s_1	15	40		0.00	1.00		1.00	0.00		1.00	1.00	7
s_2	17	36		0.07	0.73		0.73	0.07		0.73	0.80	4
s_3	20	35		0.17	0.67		0.67	0.17		0.67	0.83	4
s_4	21	33		0.20	0.53		0.53	0.20		0.53	0.73	3
s_5	24	28	(4.1)	0.30	0.20	(4.2)	0.30	0.20	(4.3)	0.30	0.50	1
s_6	29	27	$\rightarrow$	0.47	0.13	$\rightarrow$	0.47	0.13	$\rightarrow$	0.47	0.60	2
s_7	39	25		0.80	0.00		0.80	0.00		0.80	0.80	5
s_8	45	26		1.00	0.07		1.00	0.07		1.00	1.07	8
s_9	41	27		0.87	0.13		0.87	0.13		0.87	1.00	6
s_{10}	40	35		0.83	0.67		0.83	0.67		0.83	1.50	8

Comme mentionné précédemment, la sL-dominance favorise les solutions qui permettent de bien optimiser tous les objectifs. En conséquence, elle tend à éliminer les solutions extrêmes. Dans l'exemple présenté, les solutions s_5, s_6 et s_4 se situent dans la région où les deux objectifs sont presque également optimisés ; alors elles sont classées les trois meilleures (dans l'ordre) par la dominance de sLorenz. En revanche, une des solutions extrêmes s_1, classée rang 1 par Pareto, n'est pas considérée comme intéressant par sLorenz (rang 7).

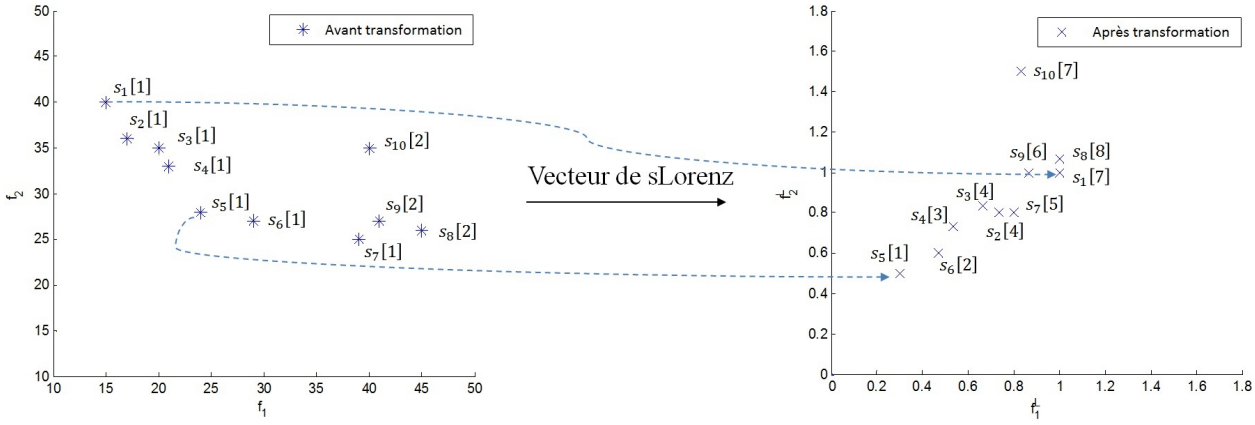

Figure 4.1 – Transformation de Pareto à sLorenz

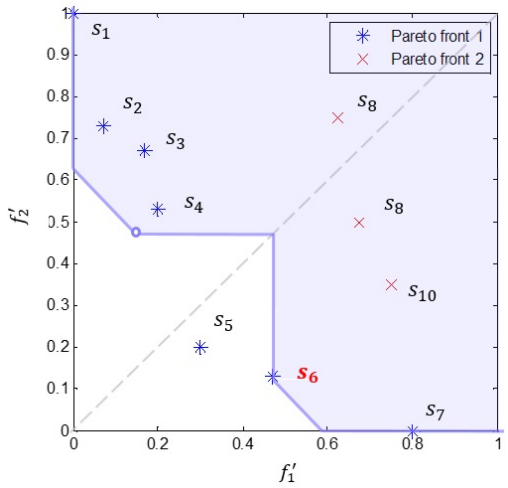

Figure 4.2 – Région dominée par s_6 dans le sens de sLorenz

De plus, pour une solution donnée, avant de calculer le vecteur de Lorenz, nous pouvons représenter sa zone de dominance (figure 4.2). Prenons l'exemple de s_6, classée $2^{\text{ième}}$ rang par sL-dominance. Nous voyons sur la figure, que s_6 domine toutes les autres solutions mise à part s_5. La région dominée est symétrique par rapport à la droite $f'_1 = f'_2$, en excluant le point symétrique de s_6 elle-même. Cette façon d'illustrer est justifiée par Dugardin, Yalaoui et Amodeo (2010) [23]. Il est clair que les solutions extrêmes sont défavorisées, comme elles sont souvent incluses dans la zone dominée d'une solution selon la dominance de sLorenz. La sL-dominance propose un classement différent de la dominance de Pareto. Elle met en priorité les solutions qui optimisent également tous les objectifs et considère les solutions extrêmes peu intéressantes, malgré leurs rang de Pareto. De ce fait, la dominance de sLorenz permet de réduire le nombre de solutions trouvées, tout en gardant de bonnes qualités sur chaque critère considéré ainsi qu'un bon compromis.

4.1.2 SC-dominance

4.1.2.1 Synthèse de la méthode

La "Controlling Dominance Area of Solutions (CDAS)" est introduit par Sato et al. (2007) [61], dans le cadre de résoudre un multi-objectif problème de sac à dos. Ils ont souligné, en prenant le NSGAII comme exemple, que la procédure de sélection au sens d'un front non-dominé dépend

largement de la diversité des solutions, la notion “dominance” n’est plus exploitée dans ce cas. A partir de cette observation, les auteurs ont proposé de modifier les régions de Pareto-dominance, et de créer des fronts non-dominés plus affinés. De ce fait, le CDAS est introduit afin de différencier les solutions non-dominées dans le sens de Pareto.

Le CDAS permet de réduire ou d’étendre les différents espaces définis par la dominance de Pareto. Pour ce faire, les solutions sont projetées sur les axes d’objectif selon un paramètre d’angles de projection. Ensuite, les solutions sont ré-classées par leurs coordonnées modifiées, selon la dominance de Pareto. Sur la figure 4.3, un exemple d’extension de la zone dominée d’une solution est présenté. Dans un bi-objectif problème de maximisation, deux angles de projection φ_1 et φ_2 sont considérés. Les solutions a, b, c sont projetées et les zones dominées par ces solutions sont élargies. Cette figure montre que $c' \prec_P a'$, $c' \prec_P b'$, alors que c est favorisé par rapport à a et b (pour des explications plus détaillées voir la suite de la section).

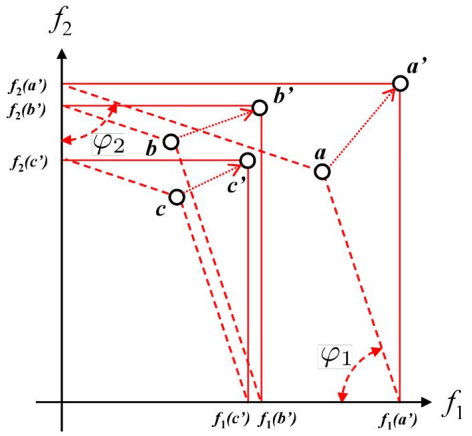

Figure 4.3 – Un exemple de CDAS [61]

Après cette première proposition, Sato et al. (2010) [62] ont amélioré leurs travaux antérieurs avec la règle “*Self*-Controlling Dominance Area of Solutions (SCDAS)”. Les fronts de Pareto sont à nouveau affinés. Contrairement au CDAS où les paramètres de contrôle sont fixés par les utilisateurs, cette nouvelle dominance calcule les paramètres nécessaires avec les solutions sur le front concerné (d’où vient “Self”). Nous adaptons la dominance “Self-Controlling” (notée par “SC-dominance” ci-après) dans nos résolutions.

4.1.2.2 Éléments de calculs

Dans l’algorithme 4, nous présentons une synthèse sur la démarche de la SC-dominance dont les explications seront développées au fur et à mesure par la suite. En complément, nous avons adopté l’exemple numérique donné au début du chapitre avec une illustration sur la figure 4.4 pour une première prévision. La solution s_5 est définie comme la “solution de référence” et l’espace SC-dominé par s_5 est mise en évidence.

Avant de commencer les calculs, les solutions sont triées dans l’ordre lexicographique selon les objectifs f_1, $f_2, \ldots$ jusqu’à f_K. Pour commencer, l’espace de solution est borné par les points d’origine O^{SC} et de repères $P_k, k \in \{1, 2, \ldots, K\}$. L’une après l’autre, les solutions jouent le rôle

Algorithm 4 SCDAS - Définir les classes

Input : F - un front de Pareto trié
Output : $C_0, C_1, ...$ - les classes selon la SC-dominance
Variables : $\Phi(s), \Omega(s)$ - vecteurs d'angles de projection d'une solution $s \in F$

```
Trouver le point d'origine et les points de repères avec les équations (4.6) and (4.7), respectivement
Pour s ∈ F faire
   c_s = 0    %Initialiser la classe par 0
Fin pour
Pour x ∈ F faire
   Trouver le vecteur d'angles Φ(x) = (φ_1(x), φ_2(x), ..., φ_K(x))    %x devient la solution de référence
   Pour y ∈ F, y ≠ x faire
      Trouver le vecteur d'angles Ω(y) = (ω_1(y), ω_2(y), ..., ω_K(y)) %y comme étant une solution non-référencée
      Calculer les coordonnées modifiées de y et obtenir sa projection y', selon la SC-dominance
   Fin pour
   Pour y ∈ F, y ≠ x faire

      Si x ≺_P y alors
         c_y = c_y + 1
      Fin si
   Fin pour
Fin pour
Pour s ∈ F faire
   C_{c_s} = C_{c_s} ∪ s    %Mettre s dans la classe correspondante
Fin pour
```

de "solution de référence", à partir de laquelle les angles de projection Φ sont calculés. Ensuite, nous considérons les autres solutions et déduisons leurs angles de projection $\Omega = (\omega_1, \omega_2, \ldots, \omega_K)$ ainsi leurs coordonnées modifiées selon la loi de sinus. Au début de la procédure, toutes les solutions appartiennent à la classe 0. Dans un front non-dominé F, nous notons pour la solution $s \in F$ sa classe c_s, qui correspond au nombre de solutions dominant s selon la SC-dominance. Après avoir examiné toutes les solutions en tant que référence, elles sont catégorisées en plusieurs classes, c'est-à-dire on trouve leur classement selon la SC-dominance.

Le point de départ de l'algorithme est de définir les points d'origine O^{SC} et des repères $P_k, k \in \{1, 2, \ldots, K\}$, comme montrés dans les équations (4.6) et (4.7).

$$O^{SC} = (f_1^{\max} + \varepsilon, f_2^{\max} + \varepsilon, \ldots, f_K^{\max} + \varepsilon) \tag{4.6}$$

$$P_k = (P_k^1, \ldots, P_k^i, \ldots, P_k^K) \text{ où } P_k^i = \begin{cases} f_k^{\min} + \varepsilon & \text{si } i \in \{1, 2, \ldots K\} \text{ et } i = k \\ f_k^{\max} + \varepsilon & \text{si } i \in \{1, 2, \ldots K\} \text{ et } i \neq k \end{cases} \tag{4.7}$$

où $f_k^{\min}$, $f_k^{\max}$ correspondent aux valeurs minimale et maximale sur l'objectif $k \in \{1, 2, \ldots, K\}$, déterminées par les solutions sur le front F. ε est une petite valeur constante pour garantir que les équations aient toujours du sens.

Dans la figure 4.4, nous reprenons l'exemple numérique et considérons seulement les solutions de rang 1 (donc $s_1 - s_7$). Lors des calculs, les approximations sont précisées jusqu'à 10^{-2},

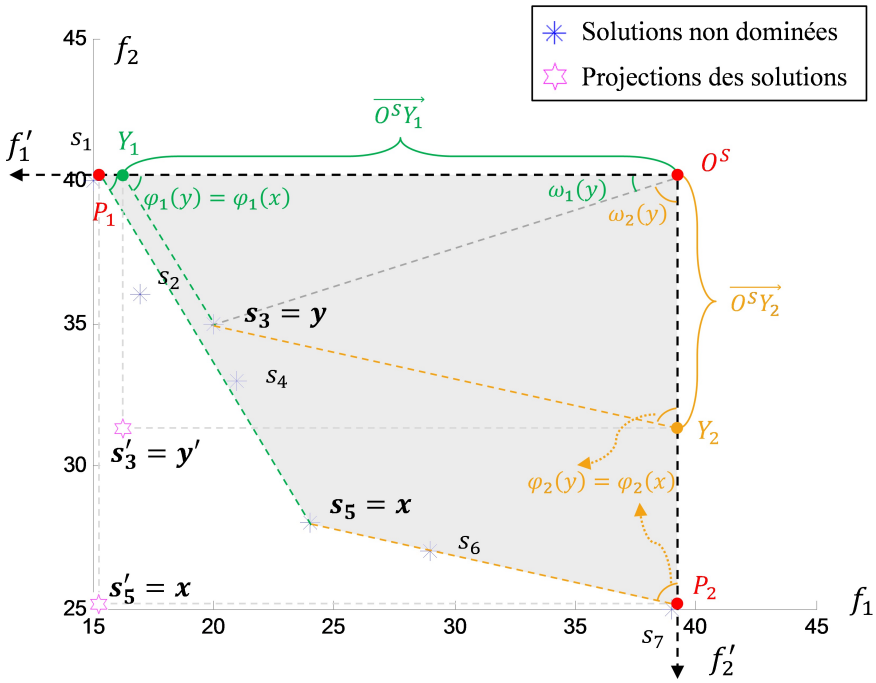

Figure 4.4 – Un exemple de SC-dominance (solution de référence = s_5, comparée avec s_3)

ainsi, nous définissons $\varepsilon = 10^{-3}$ pour assurer une influence acceptable. On obtient $O^{SC} = (39.001, 40.001)$, $P_1 = (15.001.40.001)$ et $P_2 = (39.001, 25.001)$ pour le problème bi-objectif.

En même temps, nous établissons le nouveau système de coordonnées avec :

$$\sigma_k = \overrightarrow{O^{SC}P_k}, \ \forall k \in \{1, 2, \ldots, K\} \tag{4.8}$$

Nous utilisons ensuite la loi des sinus pour trouver les projections de solutions non-référencées sur les axes $\sigma_k, \forall k \in \{1, 2, \ldots, K\}$. Dans la figure, il convient de prêter l'attention aux triangles $\triangle Y_1 O^{SC} y$ et $\triangle Y_2 O^{SC} y$.

Plus généralement, notons x la solution de référence et y la solution que l'on souhaite projeter. Pour y, notons le point de projections sur l'axe $\sigma_k (k \in \{1, 2, \ldots, K\})$ par Y_k. Pour une solution s donnée (que ce soit x ou y ou autre), le vecteur d'angles entre $\overrightarrow{P_k O^{SC}}$ et les axes $\overrightarrow{P_k s}$ est noté par $\Phi(s) = (\varphi_1(s), \varphi_2(s), \ldots, \varphi_K(s))$; celui entre $\overrightarrow{O^{SC} P_k}$ et $O^{SC} s$ font les angles $\Omega(s) = (\omega_1(s), \omega_2(s), \ldots, \omega_K(s))$.

Dans la méthode de SCDAS, la projection de la solution de référence se trouve proche du point Ideal, telle que $x' = (f_1^{\min} + \varepsilon, f_2^{\min} + \varepsilon, \ldots, f_K^{\min} + \varepsilon)$. C'est-à-dire que les angles de projection de x sont fixes. Ainsi, la question qui se pose ici est "Qu'est ce qu'on obtient si y est projetée de la même manière que x ?" Pour répondre, il suffit de garantir que $\varphi_k(y) = \varphi_k(x), \forall k \in \{1, 2, \ldots, K\}$. Par conséquent, les valeurs $\sigma_k(y)$ peuvent être obtenues, selon la relation 4.9 :

$$\sigma_k(y) = |\overrightarrow{O^{SC} Y_k}| = \frac{|\overrightarrow{O^{SC} y}| * \sin(\varphi_k(x) + \omega_k(y))}{\sin \varphi_k(x)} \ \forall k \in \{1, 2, \ldots, K\} \tag{4.9}$$

Par exemple, dans la figure 4.4, il faut considérer $\triangle Y_1 O^{SC} y$ pour trouver la projection de y sur f_1' :

$$\sigma_1(y) = \frac{|\overrightarrow{O^{SC}y}| * \sin(\varphi_1(x) + \omega_1(y))}{\sin \varphi_1(x)}$$

Finalement, nous trouverons les coordonnées de y' (la projection de y) dans les coordonnées d'origine (avec $f_k, k \in \{1, 2, \ldots, K\}$) selon l'équation suivante :

$$y'_k = f_k^{\max} + \varepsilon - \sigma_k(y),\ \forall k \in \{1, 2, \ldots, K\} \tag{4.10}$$

Par ailleurs, en appliquant ces formules sur x, on retrouve sa projection sur le point $x' = (f_1^{\min} + \varepsilon, f_2^{\min} + \varepsilon, \ldots, f_K^{\min} + \varepsilon)$.

4.1.2.3 Exemple numérique

Prenons à nouveau l'exemple numérique donné au début du chapitre, et continuons sur le cas où $x = s_5$ et $y = s_3$. Nous rappelons que :

$$\begin{cases} O^{SC} = (39.001, 40.001) \\ P_1 = (15.001.40.001) \\ P_2 = (39.001, 25.001) \end{cases} \qquad \begin{cases} x = (24, 28) \\ y = (20, 35) \end{cases}$$

On peut déduire $\Phi(x)$ (donc $Phi(s_5)$), dans un premier temps, car

$$\cos \varphi_k(x) = (\overrightarrow{P_k x} \cdot \overrightarrow{P_k O^{SC}}) \Big/ (|\overrightarrow{P_k x}| * |\overrightarrow{P_k O^{SC}}|)\ ,\ \forall k \in \{1, 2\}$$

Nous avons alors :

$$\begin{cases} cos(\varphi_1(x)) = cos(\varphi_1(s_5)) = \frac{\overrightarrow{P_1 s_5} \cdot \overrightarrow{P_1 O^{SC}}}{|\overrightarrow{P_1 s_5}| * |\overrightarrow{P_1 O^{SC}}|} = \frac{(8.999, -12.001) \cdot (24,0)}{|(8.999, -12.001)| * |(24,0)|} = 0.60 \\ cos(\varphi_2(x)) = cos(\varphi_2(s_5)) = \frac{\overrightarrow{P_2 s_5} \cdot \overrightarrow{P_2 O^{SC}}}{|\overrightarrow{P_2 s_5}| * |\overrightarrow{P_2 O^{SC}}|} = \frac{(-15.001, 2.999) \cdot (0,15)}{|(-15.001, 2.999)| * |(0,15)|} = 0.20 \end{cases}$$

$$\rightarrow \begin{cases} \cos(\varphi_1(x)) = 0.60 \\ \cos(\varphi_2(x)) = 0.20 \end{cases} \rightarrow \begin{cases} \sin(\varphi_1(x)) = 0.80 \\ \sin(\varphi_2(x)) = 0.98 \end{cases} \rightarrow \begin{cases} \varphi_1(x) = 0.93 \\ \varphi_2(x) = 1.37 \end{cases} \rightarrow \begin{cases} \varphi_1(x) = 0.30\pi \\ \varphi_2(x) = 0.44\pi \end{cases}$$

Avec la même démarche, on peut obtenir les angles $\Omega(y)$ pour n'importe quelle solution non-référencée $y \neq x, y \in F$:

$$\cos \omega_k(y) = (\overrightarrow{O^{SC} P_k} \cdot \overrightarrow{O^{SC} y}) \Big/ (|\overrightarrow{O^{SC} P_k}| * |\overrightarrow{O^{SC} y}|)\ ,\ \forall k \in \{1, 2\}$$

Prenons toujours s_3 comme exemple ($y = s_3$), alors nous avons :

$$\begin{cases} \cos(\omega_1(y)) = 0.97 \\ \cos(\omega_2(y)) = 0.25 \end{cases} \rightarrow \begin{cases} \sin(\omega_1(y)) = 0.25 \\ \sin(\omega_2(y)) = 0.97 \end{cases} \rightarrow \begin{cases} \omega_1(y) = 0.08\pi \\ \omega_2(y) = 0.42\pi \end{cases}$$

Enfin, nous calculons y', selon les équations (4.9) et (4.10) la projection de y :

$$\begin{cases} \sigma_1(y) = \frac{|\overrightarrow{O^{SC}y}|*\sin(\varphi_1(x)+\omega_1(y))}{\sin\varphi_1(x)} \\ \sigma_2(y) = \frac{|\overrightarrow{O^{SC}y}|*\sin(\varphi_2(x)+\omega_2(y))}{\sin\varphi_2(x)} \end{cases} \rightarrow \begin{cases} \sigma_1(y) = \frac{19.65*\sin(0.38\pi)}{0.80} = 22.75 \\ \sigma_2(y) = \frac{19.65*\sin(0.86\pi)}{0.98} = 8.80 \end{cases} \rightarrow \underline{y' = (16.25, 31.20)}$$

Or $x' = (f_1^{\min} + \varepsilon, f_2^{\min} + \varepsilon) = (15.001, 25.001)$, alors $x' \prec_P y'$. Ainsi, on peut en déduire :

$$\underline{x \prec_{SC} y \iff s_5 \prec_{SC} s_3}$$

Avec le même principe, nous examinons toutes les autres solutions et obtenons les résultats comme montré le tableau 4.2. Lors que x_5 est considérée comme la solution de référence, s_3 et s_4 sont SC-dominées. Ceci rejoint l'illustration dans la figure 4.4, où ces deux solutions sont incluses dans la région SC-dominée de s_5. En effet, pour avoir le classement final des solutions, s_1 - s_7 sont une

Tableau 4.2 – SC-dominance – exemple de calcul avec $x = s_5$

	f_1	f_2		σ_1	σ_2	SC-classe
s_1	15	40	s'_1	14.99	35.20	+0
s_2	17	36	s'_2	14.00	31.60	+0
s_3	20	35	s'_3	16.25	31.20	+1
s_4	21	33	s'_4	15.75	29.40	+1
$s_5 = x$	**24**	**28**	$s'_5 = x'$	**15.001**	**25.001**	-
s_6	29	27	s'_6	19.25	25.00	+0
s_7	39	25	s'_7	27.75	25.00	+0

après l'autre prises comme référence. Dans le tableau 4.3, nous présentons les résultats complets – pour une case (y, x) (ligne, colonne), nous notons l'implémentation de classe de y, en prenant x comme solution de référence.

Finalement, selon la SC-dominance, 3 classes sont identifiées – $C_0 = \{s_1, s_2, s_5, s_6, s_7\}$, $C_1 = \{s_4\}$ et $C_3 = \{s_3\}$. Avec cette règle de dominance, les solutions extrêmes sont protégées ainsi que celles qui sont proche du point Ideal.

4.1.3 mSC-dominance

4.1.3.1 Synthèse et éléments de calculs

Comme remarqué dans les deux sous-sections précédentes, la sL-dominance défavorise les solutions extrêmes alors que la SC-dominance les protège. En s'inspirant de ces deux règles, dans un problème bi-objectif, Moghaddam (2012) [48] a proposé la mSC-dominance qui élimine les solutions extrêmes à priori, tout en gardant la démarche générale de la SC-dominance. Le principe de cette règle est de garder l'idée d'étendre ou réduire les espaces de dominance, mais aussi de se concentrer sur la régions proche du point Ideal.

Tableau 4.3 – SC-dominance – exemple numérique SC-classe

$y\|x$	s_1	s_2	s_3	s_4	s_5	s_6	s_7	SC-classe
s_1	-	+0	+0	+0	+0	+0	+0	0
s_2	+0	-	+0	+0	+0	+0	+0	0
s_3	+0	+1	-	+1	+1	+0	+0	3
s_4	+0	+0	+0	-	+1	+0	+0	1
s_5	+0	+0	+0	+0	-	+0	+0	0
s_6	+0	+0	+0	+0	+0	-	+0	0
s_7	+0	+0	+0	+0	+0	+0	-	0

Pour ce faire, Moghaddam (2012) [48] ont introduit une notion de “SLOPE”. Ceci permet de former des similitudes afin de déduire les projections, comme montré dans la figure 4.5 (où X_1, X_2, X_3 et X_4 sont 4 solutions Pareto non-dominées). Cette proposition a gardé la plupart des manipulations de la SC-dominance, la différence principale réside dans le fait que la localisation du point d’origine et des points de repère. Au lieu de chercher le point Nadir, la SC-dominance propose de placer le point d’origine O^{mSC} vers le point Ideal (formulation définie ci-après). Ce changement permet d’inverser l’effet pris par la SC-dominance sur les solutions extrêmes – au lieu de les garder, la mSC-dominance tend à éliminer ces points.

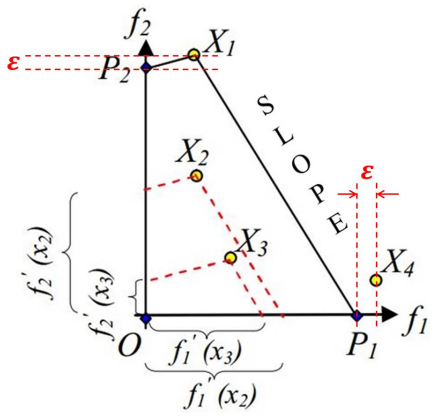

Figure 4.5 – Un extrait d’illustration apportée dans [48]

Dans nos études, nos adaptons la SC-dominance à notre problème multi-objectif. Puisque la notion de “SLOPE” est en réalité la pente d’une droite. Elle permet de simplifier les calculs pour les problèmes bi-objectifs. Mais cette simplification n’est pas conçue pour un cas avec trois objectifs. Pour ce faire, nous revenons à la formulation de Sato et al. (2010) [60], tout en gardant en pensée la mSC-dominance.

Pour présenter la mSC-dominance, nous proposons un résumé par les équations (4.11) – (4.13) pour les raisons suivantes :

- La mSC-dominance est une variante de la SC-dominance. Elle modifie les points d’origine et de repère mais garde les démarches principales de cette dernière.
- Les calculs de la mSC-dominance et SC-dominance sont relativement complexes, un résumé pourra faire ressortir les éléments importants.
- La plupart des équations concernées sont déjà expliquées dans la section 4.1.2.

$$x \prec_{mSC} y \text{ if } x \prec_P y \quad \underline{\text{ou}} \quad x' \prec_P y' \tag{4.11}$$

$$O^{mSC} = (f_1^{\min} + \varepsilon, f_2^{\min} + \varepsilon, \ldots, f_K^{\min} + \varepsilon) \tag{4.12}$$

$$P_k = (P_k^1, \ldots, P_k^i, \ldots, P_k^K) \text{ où } P_k^i = \begin{cases} f_k^{\max} + \varepsilon & \text{si } i \in \{1, 2, \ldots K\} \text{ et } i = k \\ f_k^{\min} + \varepsilon & \text{si } i \in \{1, 2, \ldots K\} \text{ et } i \neq k \end{cases} \tag{4.13}$$

$$x'_k = f_k^{\max},\ \forall k \in \{1, 2, \ldots, K\} \tag{4.14}$$

$$\sigma_k(y) = |\overrightarrow{O^{mSC}Y_k}| = \frac{|\overrightarrow{O^{mSC}y}| * \sin(\varphi_k(x) + \omega_k(y))}{\sin \varphi_k(x)},\ \forall k \in \{1, 2, \ldots, K\} \tag{4.15}$$

$$y'_k = \sigma_k(y) + f_k^{min},\ \forall k \in \{1, 2, \ldots, K\} \tag{4.16}$$

$$\varphi_k(s) = \arccos\left(\frac{\overrightarrow{P_k s} \cdot \overrightarrow{P_k O^{mSC}}}{|\overrightarrow{P_k s}| * |\overrightarrow{P_k O^{mSC}}|}\right),\ \forall k \in \{1, 2, \ldots K\}, \forall s \in \{x, y\} \tag{4.17}$$

$$\omega_k(s) = \arcsin\left(\frac{\overrightarrow{O^{mSC}P_k} \cdot \overrightarrow{O^{mSC}s}}{|\overrightarrow{O^{mSC}P_k}| * |\overrightarrow{O^{mSC}s}|}\right),\ \forall k \in \{1, 2, \ldots, K\}, \forall s \in \{x, y\} \tag{4.18}$$

Les conditions d'établir une relation de la mSC-dominance sont explicitées dans l'équation (4.11) – comme la SC-dominance, elle n'inverse pas la dominance de Pareto ; les coordonnées projetées sont calculées pour deux solutions Pareto-équivalentes et les projections sont ensuite comparées selon la dominance de Pareto. Dans les équations (4.12) et (4.13), les points d'origine et les points de repère sont définis. Les équations (4.14), (4.15) et (4.16) permettent de calculer les projections des solutions x et y, qui nécessitent les équations (4.17) et (4.18) pour calculer les angles auxiliaires φ_k et ω_k ($k \in \{1, 2, \ldots, K\}$). Remarquons également que les équations (4.14) – (4.18), et la démarche générale de la mSC-dominance ne changent pas par rapport à la SC-dominance (voir aussi l'Algorithme 4, section 4.1.2.2). Seules les équations (4.12) et (4.13) sont définies différemment, ce qui ramène aussi à une différence sur la relation entre y'_k et $\sigma_k(y)$, comme décrit dans l'équation (4.16). Dans la figure 4.6, ces changements sont mis en évidence, en considérant l'exemple numérique pris dans cette section.

4.1.3.2 Exemple numérique

Suite aux changements présentés ci-dessus, la mSC-dominance propose des classements différents que la SC-dominance.

Prenons comme exemple cette fois, $x = s_3$ en tant que solution de référence et étudions sa relation avec $y = s_5$. Dans la figure 4.6, cet exemple est illustré, où la partie grisée représente la région dominée dans le sens du mSC-dominance. En comparaison avec la figure 4.4, nous pouvons observer les deux règles de traitement des solutions extrêmes différemment – la SC-dominance les protège automatiquement alors que avec la mSC-dominance, elles sont toujours dominées.

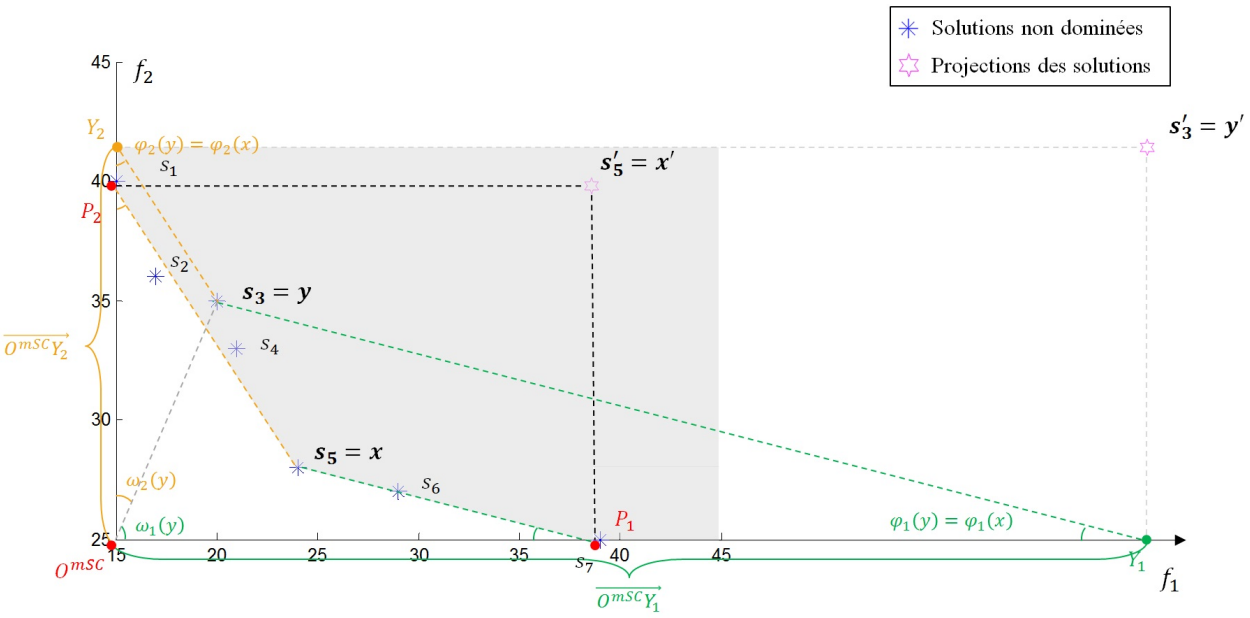

Figure 4.6 – Un exemple de mSC-dominance (solution de référence = s_5, comparée avec s_3)

Plus concrètement, selon les équations (4.11) et (4.12), l'espace de solution est encadré par :

$$\begin{cases} O^{mSC} = (14.999, 24.999) \\ P_1 = (38.999.24.999) \\ P_2 = (14.999, 39.999) \end{cases} \quad \text{et aussi} \quad \begin{cases} x = (20, 35) \\ y = (24, 28) \end{cases}$$

Les démarches ci-après rejoignent la SC-dominance. Tout d'abord, nous calculons les angles de projection entre x et les points repère d'après l'équation (4.17), telles que :

$$\begin{cases} \cos(\varphi_1(x)) = 0.98 \\ \cos(\varphi_2(x)) = 0.80 \end{cases} \rightarrow \begin{cases} \sin(\varphi_1(x)) = 0.20 \\ \sin(\varphi_2(x)) = 0.60 \end{cases} \rightarrow \begin{cases} \varphi_1(x) = 0.06\pi \\ \varphi_2(x) = 0.20\pi \end{cases}$$

Ensuite, l'équation (4.18) permet de déterminer les angles entre le point d'origine et la solution y :

$$\begin{cases} \cos(\omega_1(y)) = 0.45 \\ \cos(\omega_2(y)) = 0.89 \end{cases} \rightarrow \begin{cases} \sin(\omega_1(y)) = 0.89 \\ \sin(\omega_2(y)) = 0.45 \end{cases} \rightarrow \begin{cases} \omega_1(y) = 0.35\pi \\ \omega_2(y) = 0.15\pi \end{cases}$$

Enfin, nous trouvons les coordonnées de y' selon les équations (4.15) et (4.17) :

$$\begin{cases} \sigma_1(y) = \frac{|\overrightarrow{O^{mSC}y}| * \sin(\varphi_1(x) + \omega_1(y))}{\sin \varphi_1(x)} \\ \sigma_2(y) = \frac{|\overrightarrow{O^{mSC}y}| * \sin(\varphi_2(x) + \omega_2(y))}{\sin \varphi_2(x)} \end{cases} \rightarrow \begin{cases} \sigma_1(y) = 54.99 \\ \sigma_2(y) = 16.67 \end{cases} \rightarrow \underline{y' = (69.99, 41.67)}$$

De ce fait, nous pouvons déduire que s_5 est mSC-dominée par s_3 :

$$\begin{cases} s_3 \not\prec_P s_5 \\ s_5 \not\prec_P s_3 \\ s'_5 \prec_P s'_3 \end{cases} \Rightarrow \; s_5 \prec_{mSC} s_3$$

En généralisant les calculs, nous obtenons les résultats ci-dessous : Par rapport à ce que le

Tableau 4.4 – mSC-dominance – exemple de calcul avec $x = s_5$

	f_1	f_2		f_1	f_2	mSC-classe
s_1	15	40	s'_1	14.99	40.00	+1
s_2	17	36	s'_2	71.98	38.67	+0
s_3	20	35	s'_3	69.99	41.67	+1
s_4	21	33	s'_4	60.99	41.00	+1
$s_5 = x$	**24**	**28**	$s'_5 = x'$	**38.999**	**39.999**	-
s_6	29	27	s'_6	39.00	45.66	+1
s_7	39	25	s'_7	39.00	57.00	+1

tableau 4.3 montre, s_1, s_6 et s_7 deviennent dominées par s_5 quand on passe de la SC-dominance à la mSC-dominance.

Avec les mêmes principes, on obtient les classements finaux comme donnés dans le tableau 4.5 – la meilleure classe contient deux solutions $C_0 = \{s_2, s_5\}$; ensuite, nous avons $C_1 = \{s_4, x_6\}$; aucune solution est trouvée dans la classe 2 donc $C_2 = \varnothing$; de la même manière, on peut écrire $C_3 = \{s_3\}$, $C_4 = \varnothing$, $C_5 = \{s_1\}$ et finalement $C_6 = \{s_7\}$.

Tableau 4.5 – mSC-dominance – exemple numérique mSC-classe

	$y\|x$	s_1	s_2	s_3	s_4	s_5	s_6	s_7	mSC-classe
	s_1	-	+1	+1	+1	+1	+1	+0	5
	s_2	+0	-	+0	+0	+0	+0	+0	0
	s_3	+0	+1	-	+1	+1	+0	+0	3
	s_4	+0	+0	+0	-	+1	+0	+0	1
	s_5	+0	+0	+0	+0	-	+0	+0	0
	s_6	+0	+0	+0	+0	+1	-	+0	1
	s_7	+1	+1	+1	+1	+1	+1	-	6

La SC-dominance et la mSC-dominance se basent toutes les deux sur la modification des régions de dominance, en cherchant les projections des solutions. Ce qui permet d'affiner les fronts

Pareto-non-dominés. De plus, la SC-dominance protège systématiquement les solutions extrêmes, tandis que la mSC-dominance défavorise ces solutions. Par conséquent, la mSC-dominance propose des fronts encore plus affinés, tout en gardant les solutions plus proches du point Ideal. Cette observation est soutenue par l'exemple numérique déroulé dans cette section et la section 4.1.2.

4.1.4 MYA-dominance

4.1.4.1 Synthèse de la méthode

Inspirés par la r-dominance de Ben Said et al. (2010) [60] (2010), Moghaddam, Yalaoui et Amodeo (2012) [46] ont developpé la MYA-dominance en 2012. Chacune des deux règles de dominance se base sur une comparaison des distances Euclidiennes pondérées Deb et Sundar (2006) [20] des solutions. Ces dominances se focalisent sur les solutions mutuellement non-dominées, cela signifie que contrairement à la sL-dominance, le classement de Pareto est toujours soutenu.

Afin de calculer les distances Euclidiennes pondérées, un point de référence est nécessaire. Comme constaté dans la publication de Ben Said et al. (2010) [60], ce point de référence dépend souvent des attentes des décideurs, et il ne correspond pas forcément à une solution faisable. Dans l'ouvrage de Moghaddam, Yalaoui et Amodeo (2012) [46], le point d'origine O est choisi en tant que référence. Dans notre étude, le point Ideal $I = (f_1^{\min}, f_2^{\min}, f_3^{\min})$ est défini comme étant une représentation de ce que les décideurs préfèrent.

4.1.4.2 Éléments de calculs

Plus généralement, pour une problème de minimisation avec K critères, la distance Euclidienne pondérée d'une solution s peut être calculée comme dans l'équation (4.19).

$$D(s,I) = \sqrt{\sum_{k=1}^{K} w_k \left(\frac{f_k(s) - f_k^{\min}}{f_k^{\max} - f_k^{\min}}\right)^2}, \quad \text{with} \quad \sum_{k=1}^{K} w_k = 1 \tag{4.19}$$

où $f_k^{\min}$ et $f_k^{\max}$ dépendent de toute la génération. w_k est le poids affecté au k^{ieme} critère ; il est à défaut défini par les décideurs afin de distinguer leurs espérances sur les différents objectifs. Dans nos travaux, nous nous focalisons sur le cas où tous les critères sont également optimisés. Cela signifie que les poids sont égaux pour tous les objectifs, sachant qu'il est toujours possible de les modifier selon les demandes des décideurs. Selon les valeurs D, nous pouvons comparer s_a et s_b, deux solutions *non-dominées dans le sens du Pareto*. s_b est MYA-dominée s_a si

$$D(s_b,I) > D(s_a,I) + \eta(D^{\max} - D^{\min}), \quad \eta \in [0.1] \tag{4.20}$$

$D^{\min}$ et $D^{\max}$ sont les valeurs minimale et maximale des D parmi toutes les solutions. η est appelé le *seuil de non-dominance*. Ainsi, la MYA-dominance peut être défini, comme dans Moghaddam, Yalaoui et Amodeo (2012) [46] :

$$s_a \prec_{MYA} s_b \iff \begin{cases} s_a \prec_P s_b \qquad \underline{\textbf{ou}} & (4.21) \\ D^r(s_b, s_a, I) = \dfrac{D(s_a, I) - D(s_b, I)}{D^{\max} - D^{\min}} < -\eta & (4.22) \end{cases}$$

où $\prec_{MYA}$ représente la relation de MYA-dominance. La MYA-dominance consiste à comparer et distinguer deux solutions Pareto-équivalentes, elle est donc à mettre en oeuvre à l'intérieure d'un front de Pareto. Par conséquent, la MYA-dominance n'inverse pas le classement proposé par la dominance de Pareto : si une solution s_a domine une solution s_b dans le sens de Pareto, nous avons alors $s_a \prec_{MYA} s_b$.

Ben Said et al. (2010) [60] ont remarqué que le seuil de non-dominance η ont une influence importante sur le comportement de la r-dominance (et donc aussi sur la MYA-dominance). En effet, η peut être considéré comme étant un paramètre pour "relativiser" ou "trancher" entre les régions non-dominées selon les valeurs D.

La MYA-dominance propose, au-delà de la r-dominance, un seuil qui permet d'exclure les solutions extrêmes, dans le but de proposer aux décideurs des bons compromis sur tous les critères. Notons PS l'ensemble de solution Pareto, où la solution Q est identifiée :

$$Q \in PS \mid D(Q, I) = \max_{s \in PS} D(s, I) \tag{4.23}$$

Autrement dit, Q est celle avec la plus grande valeur D parmi toutes les solutions Pareto. Et finalement, nous pouvons définir la valeur du seuil η comme suit :

$$\eta = \frac{D(Q, I) - \max_{s' \in PS \setminus \{Q\}} D(s', I)}{D^{\max} - D^{\min}} + \varepsilon \tag{4.24}$$

Où ε est une valeur constante qui permet de protéger les solutions extrêmes au cas où ce sont les seules solutions intéressantes. Comme η est déterminé par les deux solutions de Pareto dont les valeurs D sont les plus grandes parmi toutes, la plus lointaine solution par rapport au I est systématiquement dominée dans le sens de MYA. Le paramètre η dépend de toutes les solutions prises en compte dans la population ; de ce fait, il peut varier radicalement, d'une génération à une autre.

4.1.4.3 Exemple numérique

Reprenons le même exemple numérique utilisé précédemment. Le point Ideal $I = (15.25)$ est défini par toutes les solutions. Étant donné que tous les objectifs sont aussi importants pour les

Tableau 4.6 – Numerical example – MYA-dominance

	f_1	f_2	$D(s,I)$		MYA-rank
s_1	15	40	<u>0.71</u>	$\leftarrow Q$	3
s_2	17	36	0.52		2
s_3	20	35	0.49		2
s_4	21	33	0.40		2
s_5	24	28	<u>0.25</u>	$D_{\min}$	1
s_6	29	27	0.34		1
s_7	39	25	<u>0.57</u>		3
s_8	45	26	0.71		-
s_9	41	27	0.62		-
s_{10}	40	35	<u>0.75</u>	$D_{\max}$	-

DMs, les valeurs D sont calculées selon l'équation (4.19), comme montré dans la colonne $D(s,I)$ du tableau 4.6. Ainsi, nous pouvons identifier $D_{\min} = 0.25$ et $D_{\max} = 0.75$. Par la suite, nous nous concentrons sur les solutions classées au rang 1 par la dominance de Pareto, c'est-à-dire $s_1 - s_7$. s_1 est donc défini comme Q, celle qui propose la plus grande valeur D. Après avoir exclu s_1, nous mettons l'accent sur $D(s_5, I)$ comme étant la solution la plus lointaine. Ceci nous permet de trouver η avec l'équation (4.24) :

$$\eta = \frac{D(Q,I) - \max_{s' \in PS \setminus \{Q\}} D(s',I)}{D^{\max} - D^{\min}} + \varepsilon = \frac{0.71 - 0.57}{0.75 - 0.25} + \varepsilon = 0.28 + \varepsilon$$

Lors des calculs, nous avons fixé la précision à 10^{-2} ; pour cette raison, on va prendre $\varepsilon = 10^{-3}$ dans cet exemple. De ce fait, nous obtenons $\eta = 0.281$. Les solutions $s_1 - s_7$ peuvent être donc classées selon l'équation (4.22). Par exemple, si on considère s_1 et s_2, nous avons :

$$D^r(s_1, s_2, I) = \frac{D(s_2,I) - D(s_1,I)}{D^{\max} - D^{\min}} = \frac{0.52 - 0.71}{0.75 - 0.25} = -0.38 < -\eta$$

s_1 est donc MYA-dominée par s_2. Au contraire, quand on compare le couple s_2 et s_3 :

$$\begin{cases} D^r(s_2, s_3, I) = \frac{D(s_3,I) - D(s_2,I)}{D^{\max} - D^{\min}} = \frac{0.49 - 0.52}{0.75 - 0.25} = -0.06 > -\eta \\ D^r(s_3, s_2, I) = -D(s_2, s_3, I) = 0.6 > -\eta \end{cases}$$

Par conséquent, la relation MYA-dominance ne peut pas être établie. Avec les mêmes principes, nous pouvons déduire les classements des solutions comme dans le tableau 4.6.

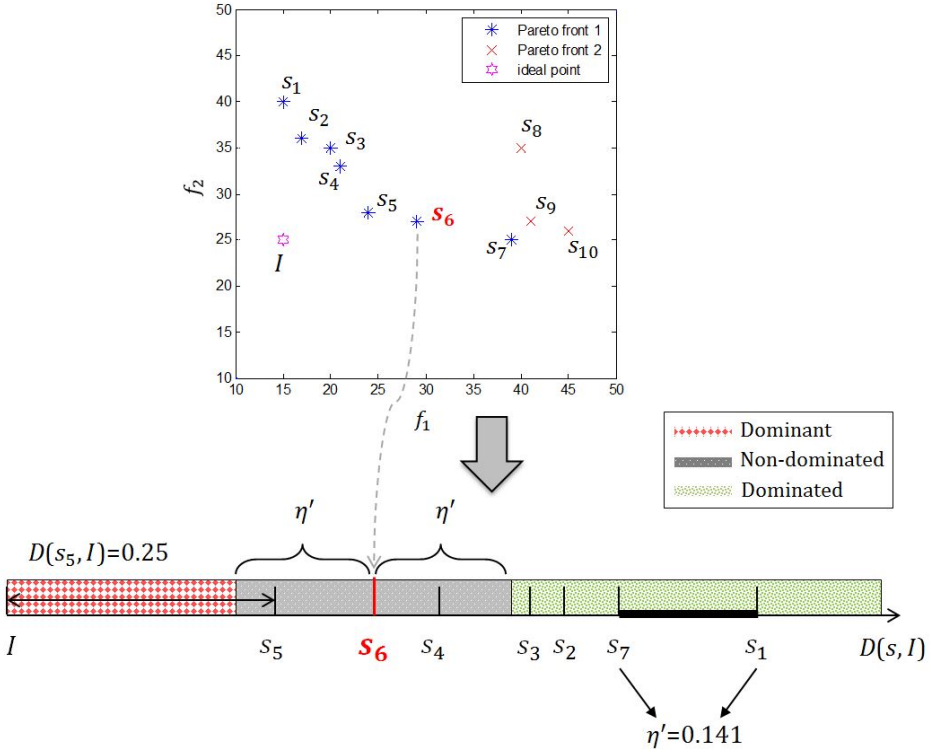

Figure 4.7 – Illustration de la MYA-domiance avec le seuil η'

4.1.4.4 Une approximation pour une meilleure illustration

Pour mieux expliciter le rôle de η, nous soulignant sur les équations (4.22) et (4.24). Si ε est suffisamment petit, nous pouvons déduire, pour deux solutions Pareto-non-dominées s_a et s_b :

$$s_a \prec_{MYA} s_b \quad \text{if} \quad D(s_a, I) - D(s_b, I) > D(Q, I) - \max_{s' \in PS \setminus \{Q\}} D(s', I) + \varepsilon \tag{4.25}$$

Cela signifie que la relation MYA-dominance est établie lorsque la différence entre $D(s_a, I)$ et $D(s_b, I)$ dépasse un certain seuil, noté par η', tel que :

$$\eta' = D(Q, I) - \max_{s' \in PS \setminus \{Q\}} D(s', I) + \varepsilon \tag{4.26}$$

Cette approximation est illustrée dans la figure 4.7. Les solutions $s_1 - s_7$ sont triées selon les valeurs D et ensuite positionnées sur l'axe $D(s, I)$ (Dans la figure, nous marquons les solutions correspondantes au lieu de donner les valeurs D). Le point Ideal I définit la plus petite distance donc 0 ; nous avons ainsi le point le plus proche s_5 avec $D(s_5, I) = 0.25$, suivi par s_6 avec $D(s_6, I) = 0.34$, etc. Le seuil η' est déterminé par s_1, la plus lointaine solution depuis I et la deuxième plus lointaine s_7, alors $\eta' = D(s_1, I) - D(s_7, I) + \varepsilon = 0.71 - 0.57 + 0.001 = 0.141$.

Prenons toujours l'exemple de s_6, les régions dominantes, non-dominée et dominée sont distinguées (voir légendes). Une zone symétrique autour de s_6, de taille $2 * \eta'$ représente l'espace non-dominé de s_6 (par exemple, s_5 est inclue). Quand une solution s n'est pas localisée dans cette zone,

si s est plus lointaine de I, alors s_6 est dominée ($s \prec_{MYA} s_6$), et sinon $s_6 \prec_{MYA} s$. Par exemple, s_2 est dominée par s_6 dans le sens de MYA car $D(s_2, I) - D(s_6, I) = 0.52 - 0.34 = 0.17 > \eta'$. De la même façon, nous pouvons identifier les différents types d'espace de dominance pour les autres solutions, et retrouver les résultats donnés dans le Tableau 4.6.

4.2 Expérimentations numériques

Dans ce chapitre, nous observons les effets des différentes propriétés de dominance sur la qualité des solutions. Notons les méthodes testées comme le suivant :

Tableau 4.7 – Notations des méthodes

	Pareto	sLorenz	SC	mSC	MYA
NSGAII	M1	M3	M5	M7	M9
NSGAIII	M2	M4	M6	M8	M10

Cependant, ces techniques risquent également de limiter la diversification de la population, car certains types de solutions sont éliminées constamment lors de l'évolution. C'est pourquoi nous mettons en œuvre par la suite la recherche locale avec la MaM pour remédier à cet inconvénient. Dans ce chapitre, nous intégrons la plus performante stratégie trouvée dans le chapitre 3 : Map5$_{ref}$, le Map5 avec la solution de référence améliorée. Notons les méthodes avec la recherche locale par M1' – M10'.

4.2.1 Tests

Les méthodes sont à nouveau comparées selon l'hypervolume et le métrique C, dont les résultats sont donnés dans les tableaux 4.8 et 4.9.

Dans le Tableau, les valeurs moyennes de l'hypervolume sont présentées, pour chaque méthode et tenant compte de tous les instances. Nous avons mis en **gras** les meilleurs résultats et souligné les moins performantes pour chaque groupe de tests. Dans la colonne "Moyenne", nous donnons la moyenne globale de chaque méthode et ensuite nous les classons dans la colonne "Rang". Dans la ligne "Meilleure amélioration", on compare la meilleure et la moins performante méthode pour le groupe correspondant. Et finalement, le nombre moyen de solutions trouvées est donné dans la colonne "N_{sol}".

Dans un premier temps, nous remarquons que la MaM permet d'améliorer l'hypervolume sur chacune des méthodes M1 – M10 (voir Tableau 4.7). Les meilleures méthodes se situent tou-

jours dans la famille “Avec MaM”, tandis que les moins performants résultats sont trouvés par les méthodes sans la MaM. Vis-à-vis de la moyenne globale, la famille “Avec MaM” améliore significativement celle sans la MaM. Cet indicateur nous permet aussi d’identifier les meilleures méthodes globale – M5’ et M10’. Elles proposent une amélioration de 34% sur les méthodes de base (M1 et M2), et un avantage de 35% sur le plus petit hypervolume (M4).

Tableau 4.8 – Comparaison de l’hypervolume : toutes méthodes

Méthodes			G30	G60	G90	G120	Moyenne	Rang	N_{sol}
Sans MaM	NSGAII	M1	0.50	0.46	0.39	0.40	0.44	16	17
	NSGAIII	M2	0.50	0.47	0.40	0.40	0.44	16	16
	L-NSGAII	M3	0.52	0.49	0.42	0.39	0.46	11	14
	L-NSGAIII	M4	0.49	0.47	0.39	0.38	0.43	20	15
	SC-NSGAII	M5	0.56	0.48	0.44	0.40	0.47	11	17
	SC-NSGAIII	M6	0.50	0.49	0.39	0.39	0.44	16	14
	mSC-NSGAII	M7	0.50	0.49	0.41	0.39	0.45	14	15
	mSC-NSGAIII	M8	0.49	0.48	0.43	0.38	0.45	14	16
	MYA-NSGAII	M9	0.50	0.46	0.41	0.40	0.44	16	16
	MYA-NSGAIII	M10	0.52	0.47	0.42	0.42	0.46	11	15
	NSGAII	M1’	0.61	0.58	0.54	0.53	0.56	6	19
Avec MaM	NSGAIII	M2’	0.62	0.59	0.54	0.52	0.57	4	18
	L-NSGAII	M3’	0.61	0.61	0.55	0.52	0.57	4	16
	L-NSGAIII	M4’	0.57	0.60	0.56	0.50	0.56	6	16
	SC-NSGAII	M5’	**0.64**	0.61	0.56	0.53	**0.59**	**1**	18
	SC-NSGAIII	M6’	0.59	0.60	**0.57**	0.49	0.56	6	15
	mSC-NSGAII	M7’	0.58	0.60	0.52	0.50	0.55	9	16
	mSC-NSGAIII	M8’	0.57	0.61	0.50	0.51	0.55	9	16
	MYA-NSGAII	M9’	0.60	**0.62**	0.55	0.54	0.58	3	17
	MYA-NSGAIII	M10’	0.62	0.60	**0.57**	**0.55**	**0.59**	**1**	16
Meilleure amélioration			0.31	0.35	0.46	0.45	0.35		

Regardons à présent la ligne “Meilleure amélioration”. Le meilleur niveau d’amélioration croit lorsque l’on augmente la taille des instances, mis à part la petite baisse par G120. Ceci est lié au paramètre Ind^{min}, qui prend la même valeur dans tous les groupes de tests. Comme il est plus difficile d’avoir des solutions convergentes pour G120 que pour les autres groupes, on envisage moins d’itérations de recherche locale pour ce premier.

D’ailleurs, tenant compte des valeurs présentées dans le Tableau 4.8, nous pouvons mettre une barrière de valeur 0.5 pour aider à évaluer les résultats. Pour les méthodes sans la MaM, cette barrière est atteinte dans peu de cas (5/40), alors que presque tous les tests (39/40) avec la MaM

réussissent à trouver un hypervolume supérieur à 0.5.

Considérons maintenant les effets des différentes alternatives de la dominance de Pareto. En regardant seules les valeurs de l'hypervolume, des améliorations sont observées mais restent assez faibles. En comparant avec la dominance de Pareto, les améliorations apportées par les autres dominances vont seulement de 0% à 5%. Les améliorations sont-elles faibles ?

En effet, comme expliqué ultérieurement et comme on peut le voir dans le tableau, le nombre de solutions trouvées avec la dominance de Pareto est supérieur à ceux trouvés avec les autres règles de dominance. Certaines dominances sont destinées à éliminer les solutions extrêmes dans l'objectif de réduire la taille du front proposé et de faciliter le travail du décideurs. Lorsque l'on considère l'hypervolume, on calcule le volume de l'espace dominé par le front de Pareto, sans tenir compte du nombre de solutions qui permettent de couvrir un tel espace. Par conséquent, cette métrique favorise naturellement les fronts avec le plus de solutions. Et dans notre cas, elle a dilué les améliorations apportées par les autres dominances.

Dans ce cas, la métrique C, présentée dans le Tableau 4.9, joue un rôle très important. Elle permet non seulement de compléter les conclusions faites selon l'hypervolume, mais aussi de donner une vision alternative et plus claire sur les relations entre les différentes règles de dominance.

Le Tableau 4.9 confirme d'abord une des conclusions obtenues avec l'hypervolume. La MaM permet d'améliorer significativement les résultats dans tous les schémas. Comme montré dans les lignes (et colonnes) correspondants à "M1" et "M2", toutes les valeurs observées sont supérieures (et respectivement inférieure) à 1. Ceci signifie que toutes les méthodes hybrides testées permettent d'améliorer les méthodes de base (M1 et M2).

On s'appuie par la suite sur les effets des dominances non-Pareto. Pour ce faire, les méthodes M1 – M10 (voir Tableau 4.7) et M1' – M10' sont divisées en 4 familles, selon leur schémas de base (NSGAII ou NSGAIII) et l'utilisation de la MaM (Avec MaM ou Sans MaM). Elles sont ensuite comparées à l'intérieure des familles. Cette façon de dégrouper les méthodes permet de voir tout de suite les effets des différentes dominances alternatives. Dans le Tableau 4.9, les comparaisons d'intérêt sont marquées par des cellules grisées. D'ailleurs, nous adoptons le même technique de graphique utilisé dans le Chapitre 3, comme montré dans la Figure 4.8. Comme on peut le voir, dans chaque famille, la sL-dominance, la SC-dominance et la MYA-dominance réussissent à proposer des fronts non-dominés de meilleure qualité que la dominance de Pareto. Parmi les 4 familles, la MYA-dominance est soulignée 3 fois et la SC-dominance est soulignée une fois.

Reprenons le Tableau 4.9. Suite à la Figure 4.8, nous mettons en évidence surtout les résultats concernant M6, M9, M9' et M10'. Parmi ces 4 méthodes, M10' s'est clairement distinguée. Elle permet non seulement d'amélioration significativement les méthodes de base, mais aussi propose

des solutions de meilleures qualités que toutes les autres méthodes. Plus généralement, son avantage reste remarquable lorsque l'on considère toutes méthodes mélangées.

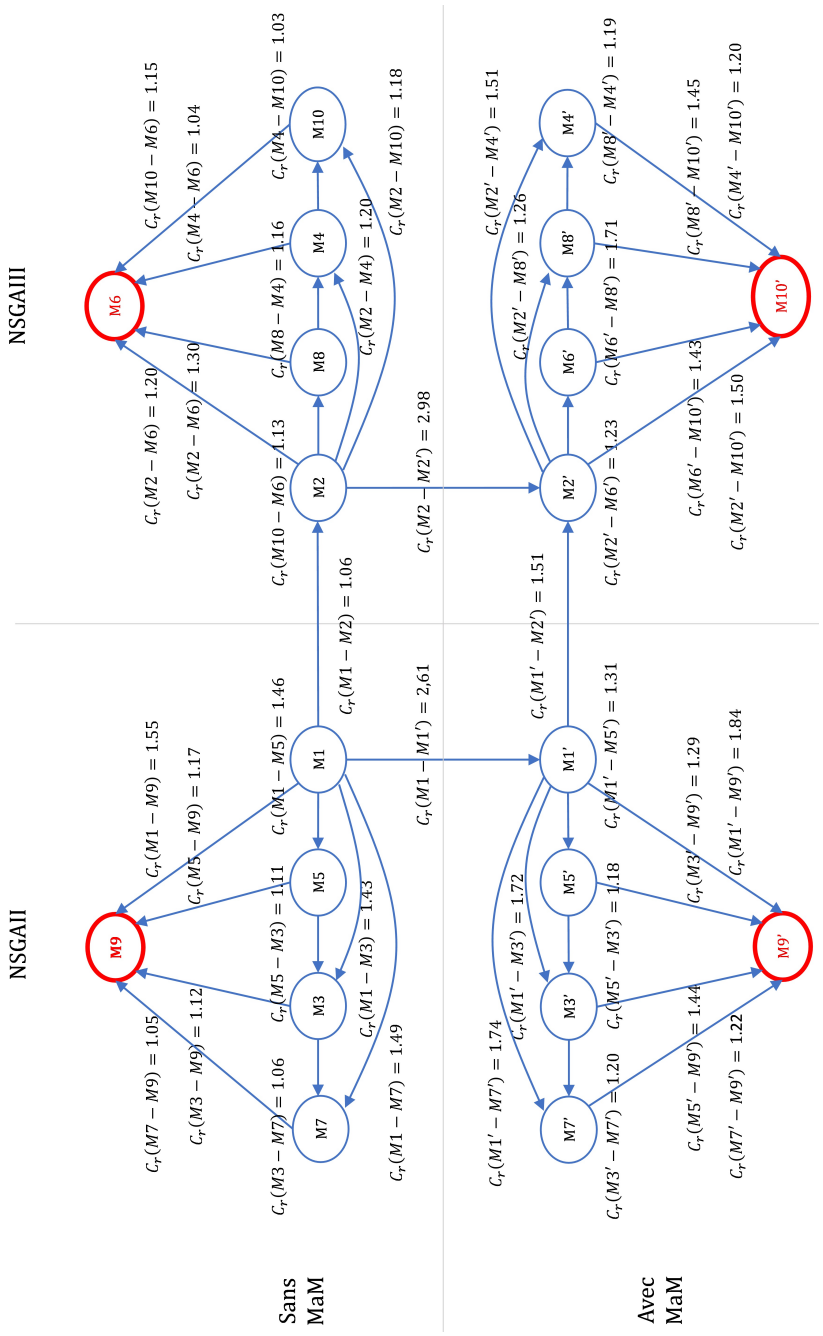

Figure 4.8 – Comparaison de la métrique C - graphique

Tableau 4.9 – Comparaison métrique C : résultats complets

C_r	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M1'	M2'	M3'	M4'	M5'	M6'	M7'	M8'	M9'	M10'
M1	-	1.06	1.43	1.73	1.46	*1.73*	1.84	1.63	*1.55*	1.49	2.61	3.95	3.42	4.75	2.62	3.51	3.68	3.47	*4.21*	**5.67**
M2	0.95	-	1.12	1.20	0.99	*1.30*	1.54	1.51	*1.08*	1.18	1.82	3.02	2.43	3.30	2.06	3.45	2.69	2.79	*2.61*	**4.10**
M3	0.71	0.94	-	1.23	0.99	1.27	1.11	0.95	1.12	1.12	1.70	2.28	2.22	2.94	2.12	3.08	2.97	2.76	2.76	3.98
M4	0.62	0.89	0.90	-	0.80	1.04	0.99	0.91	0.88	1.03	1.42	2.55	2.34	3.75	1.88	3.02	2.36	3.14	2.41	4.08
M5	0.75	1.07	1.11	1.30	-	1.34	1.03	1.02	1.19	1.21	1.53	3.16	2.33	3.49	2.01	3.07	4.17	2.82	3.09	4.47
M6	*0.60*	*0.80*	0.84	1.00	0.78	-	1.16	0.87	*0.89*	0.96	1.46	1.97	2.16	2.62	1.67	2.60	3.03	2.92	*1.95*	*3.55*
M7	0.57	0.69	0.93	1.02	0.99	0.88	-	0.85	0.89	0.92	1.39	1.64	2.17	2.59	1.88	2.35	2.84	2.04	3.14	3.31
M8	0.63	0.73	1.07	1.12	0.99	1.20	1.20	-	1.17	1.04	1.50	2.85	2.28	3.43	1.92	2.84	3.60	2.22	3.39	3.75
M9	*0.70*	*1.00*	0.98	1.17	0.85	*1.20*	1.16	0.86	-	1.12	1.39	2.75	2.21	3.57	1.85	2.73	2.63	2.51	*2.76*	*3.66*
M10	0.67	0.91	0.91	1.04	0.84	1.15	1.11	0.98	0.93	-	1.41	2.50	2.20	3.41	1.75	2.74	2.29	2.31	2.35	3.85
M1'	0.39	0.61	0.72	0.73	0.70	0.81	0.75	0.69	0.75	0.78	-	1.51	1.72	2.40	1.31	1.59	1.56	1.57	1.84	2.38
M2'	0.27	0.33	0.47	0.41	0.36	0.55	0.36	0.36	0.39	0.45	0.68	-	0.89	1.51	0.79	1.23	1.15	1.06	1.10	1.50
M3'	0.31	0.45	0.49	0.53	0.50	0.52	0.47	0.45	0.51	0.47	0.69	1.20	-	1.47	0.86	1.22	1.20	1.12	1.29	1.59
M4'	0.25	0.33	0.35	0.34	0.33	0.39	0.39	0.30	0.31	0.34	0.49	0.73	0.71	-	0.55	0.88	0.77	0.82	0.78	1.20
M5'	0.39	0.52	0.50	0.58	0.53	0.64	0.55	0.53	0.59	0.59	0.89	1.34	1.18	1.84	-	1.34	1.57	1.48	1.44	2.03
M6'	0.31	0.39	0.41	0.46	0.45	0.47	0.43	0.35	0.45	0.47	0.68	1.04	0.97	1.40	0.89	-	1.65	1.71	1.17	1.43
M7'	0.28	0.39	0.35	0.45	0.36	0.34	0.37	0.30	0.41	0.48	0.66	0.92	0.86	1.34	0.67	0.64	-	0.82	1.22	1.48
M8'	0.30	0.38	0.38	0.35	0.38	0.36	0.52	0.49	0.42	0.45	0.67	0.97	0.95	1.35	0.72	0.63	1.27	-	1.30	1.52
M9'	*0.26*	*0.39*	0.40	0.44	0.37	*0.52*	0.32	0.30	*0.39*	0.46	0.58	0.92	0.87	1.31	0.70	0.97	0.84	0.78	-	*1.35*
M10'	**0.21**	**0.32**	0.30	0.29	0.27	*0.34*	0.31	0.27	*0.32*	0.34	0.43	0.69	0.74	0.96	0.58	0.79	0.74	0.72	*0.77*	-

Dans la Figure 4.9, on compare les fronts approchés des 3 méthodes représentantes, dont M2 pour les méthodes de base, M6 pour les méthodes sans MaM et M10' pour les méthodes avec MaM (meilleur de chaque type selon la métrique C). Il est montré dans la figure, que M2 est bloquée dans la zone d'optimum local, alors que M6 et M10' ont réussi à s'en sortir et se rapprocher encore plus du point Ideal. D'ailleurs, M6 et M10' permettent de réduire le nombre de solutions sur le front de Pareto approché comme elles éliminent les solutions extrêmes.

En résumé, la métrique C a complété les conclusions que l'on a obtenu avec l'hypervolume car avec ce dernier, les performances de M5' et M10' sont équivalentes. Cependant, avec la métrique C, on voit que c'est M8 qui propose les meilleurs fronts non-dominés. Cela signifie que la méthode hybride prenant compte du NSGAIII, de la MYA-dominance et la MaM est la plus performante avec notre protocole de test sur notre problème.

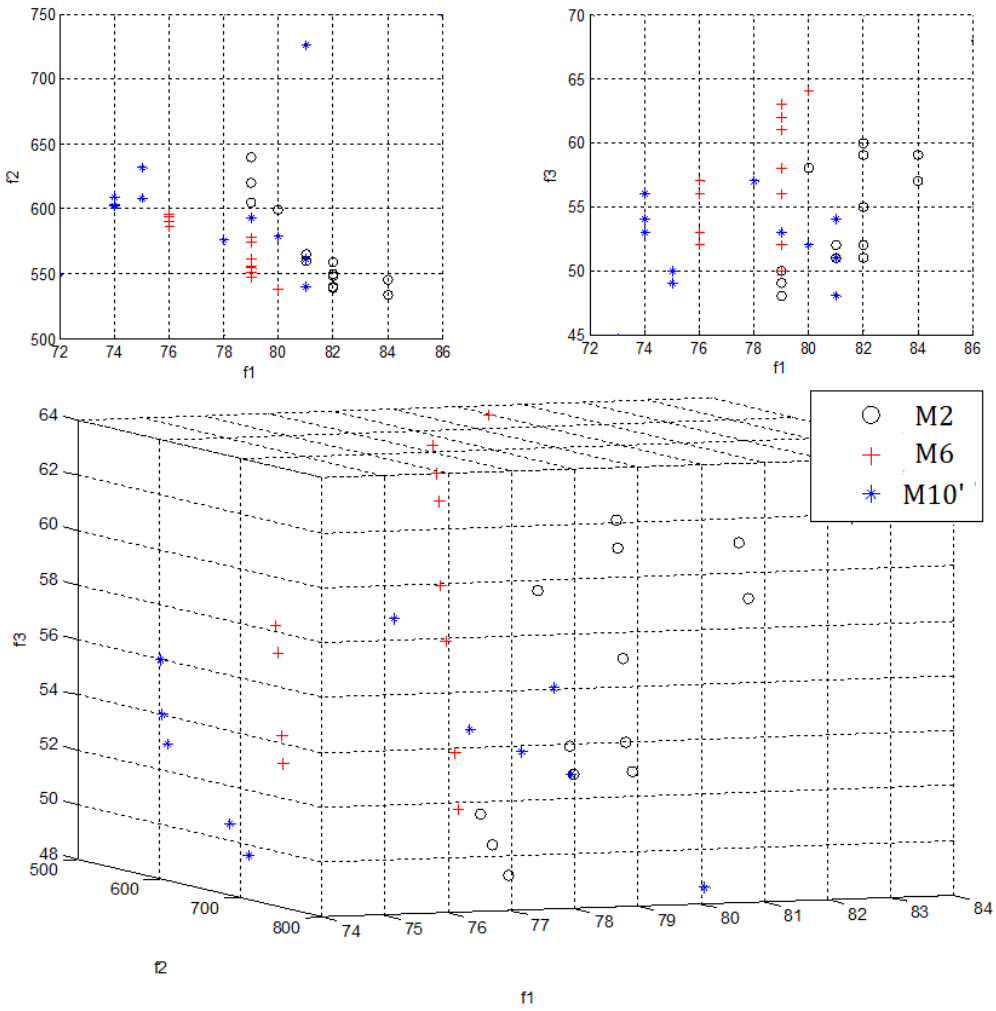

Figure 4.9 – Comparaison des fronts approchés

4.2.2 Conclusion

Dans ce chapitre, nous avons considéré un cas spécial où le décideur souhaite réduire le nombre de solutions. A ce propos, nous intégrons plusieurs règles de dominance comme alternatives de la dominance de Pareto. Plus précisément, il s'agit d'une variante de L-dominance (Kostreva et Ogryczak (1999) [42]), SC-dominance (Sato et al. (2007) [61]) et sa variante mSC-dominance (Moghaddam, Yalaoui et Amodeo (2012) [48]) et MYA-dominance (Moghaddam, Yalaoui et Amodeo (2015) [47]). De plus, la MaM présentée dans le chapitre 3 est mise en œuvre pour améliorer la qualité des solutions.

Les résultats sont évalués selon l'hypervolume et la métrique C. Chacune des métriques montre les améliorations apportées par la MaM ainsi que par les différentes règles de dominance. Nous remarquons dans un premier temps, que la MaM permet d'augmenter significativement l'hypervolume (jusqu'à 46%). Les améliorations grâce aux différentes dominances, quant à elles sont "diluées" par le nombre réduit des solutions vis-à-vis de l'hypervolume. Cependant leurs avantages importants sont mis en évidence en considérant la métrique C.

Tenant compte des deux métriques, la meilleure méthode hybridée trouvée dans ce chapitre, voire dans cette thèse, est constituée par le NSGAIII, la MYA-dominance et la MaM (Map5$_{ref}$).

Conclusion et perspectives

Cette thèse porte sur la résolution d'un problème d'ordonnancement multi-objectif avec contraintes de ressources. Trois objectifs sont pris en compte, dont le makespan, le retard total des tâches et l'équilibrage d'allocation de ressources. Des approches exactes et des métaheuristiques sont mises en œuvre pour trouver les meilleures solutions. Les méthodes exactes garantissent l'optimalité des solutions, tandis que les méthodes approchées améliorent le temps d'exécution avec une bonne qualité des résultats.

Dans le premier chapitre, nous avons montré l'intérêt croissant porté aux problèmes de type RCPSP au cours des dernières dizaines d'années. A l'aide d'une classifications des variantes du RCPSP, nous avons dégagé les principales tendances développées dans la littérature, ainsi que les problèmes qui ont le plus intéressé les chercheurs. Nous avons aussi mis en évidence un manque de travaux concernant la résolution du RCPSP dans le cas multi-objectif, contrairement au cas mono-objectif. Une revue détaillée des travaux concernant le RCPSP multi-objectif est proposée. Les études sont comparées selon les objectifs choisis et les méthodes de résolution. En effet, peu de travaux sont consacrés à la résolution exacte du RCPSP multi-objectif. D'ailleurs, la plupart des travaux concernent des problème bi-objectif, les critères les plus étudiés restent le makespan, le coût et le retard du projet. Ceci nous a permis non seulement d'identifier les objectifs les plus attractifs pour le sujet du type RCPSP, mais aussi d'apercevoir un manque d'exploration des critères liés aux ressources. Suite à ce constat, nous avons d'abord mis en œuvre des méthodes exactes pour le problème bi-objectif, et ensuite introduit l'équilibrage d'allocation des ressource dans la littérature du RCPSP et résolu le problème avec trois objectifs.

De plus, une description synthétique des méthodes les plus référencées dans la littérature, y compris les méthodes exactes et approchées, est réalisée. A la fin de ce chapitre, nous mettons l'accent sur l'évaluation des performances pour les problèmes multi-objectifs. Après une brève description des mesures le plus couramment utilisées, nous donnons les détails sur les métriques qui seront utilisées dans la suite de ce mémoire.

Le Chapitre 2 est consacré à un problème bi-objectif, où sont minimiser le makespan et le retard total des tâches. Deux formulations mathématiques sont d'abord présentées afin de modéliser le problème bi-objecif de manière formelle, et d'établir une base de la résolution exacte. En ce qui concerne les deux modèles mathématiques, le premier modèle repose sur une formulation plus intuitive et directe tandis que le modèle 2 est plus ciblé et performant.

Pour résoudre le problème bi-objectif, nous avons mis en œuvre la méthode à deux phases (TPM) et la méthode de partitionnement en parallèle (PPM). Elles sont ensuite comparées sur des instances de petite taille. Sachant que chacune des deux méthodes nécessite de résoudre une série de problèmes mono-critère dont les objectifs sont pondérés, nous avons adopté le modèle mathématique 2 pour ces tests. Les résultats obtenus par ces deux méthodes se confirment, tout en sachant que la TPM a une efficacité plus stable, alors que la performance de la PPM dépend du paramètre de parallélisme. Sur tous les tests effectués, les deux méthodes réussissent à résoudre les instances jusqu'à 15 tâches en temps limité (1800 secondes) et rencontrent des difficultés sur les instances à 20 tâches. De ce fait, nous adoptons le NSGAII, un algorithme génétique destiné aux problèmes multi-objectifs, afin d'améliorer le temps d'exécution. Les fronts de Pareto approchés sont ensuite comparés aux résultats optimaux selon deux métriques d'évaluation. Les résultats montrent l'avantage de la résolution approchée en terme de temps de calcul, ainsi que son compromis sur la qualité des solutions.

Dans le chapitre suivant, nous introduisons le troisième objectif considéré dans cette thèse : l'équilibrage d'allocation des ressources. Ce problème tri-objectif est traité dans la suite de la thèse. Dans un premier temps, nous avons adapté le NSGAII utilisé dans le chapitre précédent. De même, le NSGAIII, une version améliorée du NSGAII, est aussi implémenté pour résoudre le problème à trois objectifs. Pour comparer l'efficacité de ces deux algorithmes, nous avons créé des instances pour des problèmes de petite, moyenne et grande taille qui varient de 30 à 120 tâches. Les expériences montrent que le NSGAIII améliore à des niveaux différents les résultats du NSGAII dans tous les groupes de tests.

Au delà des schémas d'origine de ces deux méthodes, nous intégrons une recherche locale basée sur la technique de Mapping [4] pour améliorer la qualité des solutions. Destinée à une meilleure exploration de l'espace de recherche, la méthode de Mapping (MaM) est premièrement proposée dans les travaux de Auturoi et al. [4] pour résoudre un problème Job-Shop flexible. Nous adoptons d'abord cette méthode dans nos algorithmes génétiques. Cinq stratégies de Mapping sont implémentées et comparées. Le Map5, qui considère à la fois l'intensification et la diversification des solutions, se distingue des autres stratégies par sa forte efficacité. Il permet une amélioration jusqu'à 31% par rapport aux méthodes de base pour certaines instances. Avec la stratégie de

Mapping choisie, nous modifions la méthode proposée dans [4] et proposons une solution de référence plus adaptée à notre problème. Comme la solution de référence est considérée comme le point d'origine du Mapping, l'efficacité de la MaM dépend fortement de ce choix. Les expériences montrent une amélioration jusqu'à 24% par la nouvelle solution de référence, tenant compte de toutes les instances mélangées. Par conséquent, lorsque la MaM est utilisée dans la suite, nous considérons le Map5$_{ref}$ avec la solution de référence améliorée.

Le Chapitre 4 traite, quant à lui, un problème plus spécifique. Face à un grand nombre de choix équivalents, les décideurs se trouvent souvent dans une situation où la décision reste difficile à prendre. Il nous est donc demandé de réaliser une sélection plus fine pour réduire le nombre de solutions et de proposer aux décideurs une aide plus adaptée à leur besoin. Pour ce faire, nous avons intégré plusieurs règles de dominance autres que la dominance de Pareto, dans l'objectif de différencier les solutions qui sont Pareto-équivalentes. Plus précisément, nous considérons la dominance de Lorenz (L-dominance) [42], la dominance "Self-Controlling" (SC-dominance) [60], sa version modifiée (m-SC-dominance) [47] [48], ainsi que la dominance proposée par Moghaddam, Yalaoui et Amodeo (MYA-dominance) [47] [48]. Ces dominances non-conventionnelles sont intégrées dans les algorithmes génétiques et leur permettent de trouver les fronts de Pareto plus affinés.

Ainsi, nous obtenons des méthodes hybridées à deux niveaux : premièrement, les algorithmes génétiques sont hybridés avec les règles de dominances non-conventionnelles ; deuxièmement, la recherche locale est couplée pour améliorer la qualité des solutions. L'efficacité des différentes règles de dominance est d'abord prouvée par une comparaison par familles. Ces règles permettent de réduire le nombre de solutions tout en assurant une meilleure qualité des fronts de Pareto approchés. Lorsque l'on compare entre elles les méthodes hybridées, toutes familles confondues, la MaM est à nouveau la plus efficace. Ceci nous permet de conclure que la meilleure performance est réalisée par l'hybridation entre le NSGAIII, la MaM (Map5$_{ref}$) et la MYA-dominance.

Dans cette thèse, nous avons considéré un problème du type RCPSP multi-objectif. Nous avons d'abord intégré la résolution exacte des problèmes bi-objectifs dans la littérature du RCPSP et puis introduit un nouveau critère concernant l'utilisation de ressource. Après avoir déterminé la limite des méthodes exactes, on s'appuie sur la résolution approchée qui demande beaucoup moins de temps d'exécution. Afin d'améliorer la qualité de solution, nous avons utilisé une recherche locale avec la technique Mapping très récente [4] (2016). Cette méthode est premièrement adaptée et ensuite améliorée pour résoudre notre problème. Comme une deuxième hybridation, nous avons adopté plusieurs règles de dominances afin d'affiner les fronts de Pareto approchés. Ceci nous a

permis de réduire le nombre de solutions selon le besoin des décideurs, ainsi d'améliorer la qualité des solutions. En conclusion, les contributions principales de cette thèse portent d'une part sur la résolution exacte d'un problème RCPSP bi-objectif, d'autre part sur l'amélioration des solutions par des différentes techniques d'hybridation d'un problème multi-objectif spécifique qui n'avait jamais été étudié dans la littérature.

Ce travail a donné lieu à la publication de trois articles de conférences internationales avec comité de lecture [79] [83] [82] et deux articles de revues internationales [80] [81].

Plusieurs pistes de recherche seraient pertinents à poursuivre : tout d'abord, nous envisageons d'appliquer les méthodes développées dans cette thèse sur les autres problèmes du type RCPSP comme le multi-mode RCPSP et le multi-projet RCPSP. Il serait d'ailleurs très intéressant de considérer un problème d'investissement des ressources. Ce problème s'approprie la formulation générale du RCPSP mais il considère comme l'objectif le coût généré par les ressources, et le makespan devient une contrainte.

En terme de méthodes, nous pouvons envisager aussi d'hybrider la méthode MaM et les règles de dominances avec d'autres métaheuristiques telles que la recherche taboue et l'essaim particulaires. Cela nous permettrait de comparer les résultats obtenus dans cette thèse et d'explorer la compatibilité parmi les différentes méthodes de résolution et techniques d'amélioration. D'ailleurs, comme la technique de Mapping propose une unique identification pour chaque solution, on pourra la transformer en règle d'élitisme dans un algorithme génétique. Contrairement aux NSGAII et NSGAIII où la sélection est basée sur la dominance de Pareto et les valeurs objectives, une sélection avec la MaM choisit, par construction, les solutions selon leur structures.

Enfin, comme une des plus importantes motivations de nos travaux, nous pouvons considérer une application industrielle et résoudre les instances dans la vie réelle. Malgré que le RCPSP soit considéré comme un sujet proche de l'industrie, il manque des travaux traitant les vraies instances dans la littérature. Ceci pourrait être due à la contrainte dure sur le temps d'exécution auprès des industries, car le RCPSP est un problème NP-difficile au sens fort. Il nous sera donc demandé de perfectionner sur les compétences de calculs de nos algorithmes, tout en garantissant les qualité des solutions.

Bibliographie

[1] Babak Abbasi, Shahram Shadrokh, and Jamal Arkat. Bi-objective resource-constrained project scheduling with robustness and makespan criteria. *Applied mathematics and computation*, 180(1) :146–152, 2006.

[2] Mohammad A Al-Fawzan and Mohamed Haouari. A bi-objective model for robust resource-constrained project scheduling. *International Journal of production economics*, 96(2) :175–187, 2005.

[3] Christian Artigues, Philippe Michelon, and Stéphane Reusser. Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research*, 149(2) :249–267, 2003.

[4] J. Autuori, F. Hnaien, and F. Yalaoui. A mapping technique for better solution exploration : NSGA-II adaptation. *Journal of Heuristics*, 22(1) :89–123, 2016.

[5] Tonius Baar, Peter Brucker, and Sigrid Knust. *Tabu Search Algorithms and Lower Bounds for the Resource-Constrained Project Scheduling Problem*, pages 1–18. Springer US, Boston, MA, 1999.

[6] Francisco Ballestín, Vicente Valls, and Sacramento Quintanilla. *Due Dates and RCPSP*, pages 79–104. Springer US, Boston, MA, 2006.

[7] Sanghamitra Bandyopadhyay, Sriparna Saha, Ujjwal Maulik, and Kalyanmoy Deb. A simulated annealing-based multiobjective optimization algorithm : Amosa. *IEEE transactions on evolutionary computation*, 12(3) :269–283, 2008.

[8] Lucio Bianco and Massimiliano Caramia. A new formulation for the project scheduling problem under limited resources. *Flexible Services and Manufacturing Journal*, 25(1) :6–24, 2013.

[9] J. Blazewicz, J.K. Lenstra, and A.H.G.Rinnooy Kan. Scheduling subject to resource constraints : classification and complexity. *Discrete Applied Mathematics*, 5(1) :11 – 24, 1983.

[10] Fayez F Boctor. Resource-constrained project scheduling by simulated annealing. *International Journal of Production Research*, 34(8) :2335–2351, 1996.

[11] KLEIN Bouleimen and HOUSNI Lecocq. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research*, 149(2) :268–281, 2003.

[12] Peter Brucker, Andreas Drexl, Rolf Möhring, Klaus Neumann, and Erwin Pesch. Resource-constrained project scheduling : Notation, classification, models, and methods. *European Journal of Operational Research*, 112(1) :3 – 41, 1999.

[13] Peter Brucker, Sigrid Knust, Arno Schoo, and Olaf Thiele. A branch and bound algorithm for the resource-constrained project scheduling problem1. *European Journal of Operational Research*, 107(2) :272 – 288, 1998.

[14] Jacques Carlier and Emmanuel Néron. On linear lower bounds for the resource constrained project scheduling problem. *European Journal of Operational Research*, 149(2) :314–324, 2003.

[15] Jacques Carlier and Emmanuel Néron. Computing redundant resources for the resource constrained project scheduling problem. *European Journal of Operational Research*, 176(3) :1452–1463, 2007.

[16] Piotr Czyżżak and Adrezej Jaszkiewicz. Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7(1) :34–47, 1998.

[17] de Sousa, Ernando Gomes, Santos, Andréa Cynthia, and Aloise, Dario José. An exact method for solving the bi-objective minimum diameter-cost spanning tree problem. *RAIRO-Oper. Res.*, 49(1) :143–160, 2015.

[18] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I : Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4) :577–601, Aug 2014.

[19] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2) :182–197, Apr 2002.

[20] K. Deb and J. Sundar. Reference point based multi-objective optimization using evolutionary algorithms. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 635–642, New York, NY, USA, 2006. ACM.

[21] Erik Demeulemeester and Willy Herroelen. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science*, 38(12) :1803–1818, 1992.

[22] C. Dhaenens, J. Lemesre, and E.G. Talbi. K-ppm : A new exact method to solve multi-objective combinatorial optimization problems. *European Journal of Operational Research*, 200(1) :45 – 53, 2010.

[23] F. Dugardin, F. Yalaoui, and L. Amodeo. New multi-objective method to solve reentrant hybrid flow shop scheduling problem. *European Journal of Operational Research*, 203(1) :22 – 31, 2010.

[24] Parviz Ghoddousi, Ehsan Eshtehardian, Shirin Jooybanpour, and Ashtad Javanmardi. Multi-mode resource-constrained discrete time–cost-resource optimization in project scheduling using non-dominated sorting genetic algorithm. *Automation in Construction*, 30 :216–227, 2013.

[25] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[26] Walter J Gutjahr. Bi-objective multi-mode project scheduling under risk aversion. *European Journal of Operational Research*, 246(2) :421–434, 2015.

[27] YV HAIMES YV, LS LASDON LS, and DA WISMER DA. On a bicriterion formation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-1(3) :296–297, 7 1971.

[28] Sönke Hartmann and Dirk Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1) :1 – 14, 2010.

[29] Sönke Hartmann. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL)*, 45(7) :733–750, 1998.

[30] Sönke Hartmann. A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics (NRL)*, 49(5) :433–448, 2002.

[31] S. Jiang, Y. S. Ong, J. Zhang, and L. Feng. Consistencies and contradictions of performance metrics in multiobjective optimization. *IEEE Transactions on Cybernetics*, 44(12) :2391–2404, Dec 2014.

[32] FS Kazemi and R Tavakkoli-Moghaddam. Solving a multi-objective multi-mode resource-constrained project scheduling problem with particle swarm optimization. *International Journal of Academic Research*, 3(1) :103–110, 2011.

[33] Somayeh Khalili, Amir Abbas Najafi, and Seyed Taghi Akhavan Niaki. Bi-objective resource constrained project scheduling problem with makespan and net present value criteria : two meta-heuristic algorithms. *The International Journal of Advanced Manufacturing Technology*, 69(1-4) :617–626, 2013.

[34] Robert Klein. *Scheduling of resource-constrained projects*, volume 10. Springer Science & Business Media, 1999.

[35] Rainer Kolisch and Sönke Hartmann. Heuristic algorithms for solving the resource-constrained project scheduling problem : Classification and computational analysis. Technical report, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, 1998.

[36] Rainer Kolisch and Arno Sprecher. Psplib - a project scheduling problem library : Or software - orsep operations research software exchange program. *European Journal of Operational Research*, 96(1) :205 – 216, 1997.

[37] Rainer Kolisch, Arno Sprecher, and Andreas Drexl. Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science*, 41(10) :1693–1703, 1995.

[38] Oumar Koné, Christian Artigues, Pierre Lopez, and Marcel Mongeau. Event-based milp models for resource-constrained project scheduling problems. *Computers & Operations Research*, 38(1) :3–13, 2011.

[39] Michael M Kostreva, Włodzimierz Ogryczak, and Adam Wierzbicki. Equitable aggregations and multiple criteria analysis. *European Journal of Operational Research*, 158(2) :362–377, 2004.

[40] Olivier Lambrechts, Erik Demeulemeester, and Willy Herroelen. A tabu search procedure for developing robust predictive project schedules. *International Journal of Production Economics*, 111(2) :493–508, 2008.

[41] J. Lemesre, C. Dhaenens, and E.G. Talbi. Parallel partitioning method (ppm) : A new exact method to solve bi-objective problems. *Computers & Operations Research*, 34(8) :2450 – 2462, 2007.

[42] M. M. Kostreva and W. Ogryczak. Linear optimization with multiple equitable criteria. *RAIRO-Oper. Res.*, 33(3) :275–297, 1999.

[43] Jorge Jose de Magalhaes Mendes, José Fernando Gonçalves, and Mauricio GC Resende. A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & Operations Research*, 36(1) :92–109, 2009.

[44] Marek Mika, Grzegorz Waligora, and Jan Wkeglarz. Simulated annealing and tabu search for multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models. *European Journal of Operational Research*, 164(3) :639–668, 2005.

[45] Aristide Mingozzi, Vittorio Maniezzo, Salvatore Ricciardelli, and Lucio Bianco. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Management science*, 44(5) :714–729, 1998.

[46] A. Moghaddam, F. Yalaoui, and L. Amodeo. *Multi-Objective Scheduling Problems for Re-Entrant Shops*, pages 271–293. Atlantis Press, Paris, 2012.

[47] A. Moghaddam, F. Yalaoui, and L. Amodeo. Efficient meta-heuristics based on various dominance criteria for a single-machine bi-criteria scheduling problem with rejection. *Journal of Manufacturing Systems*, 34 :12 – 22, 2015.

[48] Atefeh Moghaddam. *Production scheduling : unavailabitlity of resources*. PhD thesis, 2012. Thèse de doctorat dirigée par Amodeo, Lionel et Yalaoui, Farouk Optimisation et Sûreté des Systèmes Troyes 2012.

[49] Atefeh Moghaddam, Farouk Yalaoui, and Lionel Amodeo. *An Efficient Meta-heuristic Based on Self-control Dominance Concept for a Bi-objective Re-entrant Scheduling Problem with Outsourcing*, pages 467–471. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[50] T. Okabe, Y. Jin, and B. Sendhoff. A critical survey of performance indices for multi-objective optimisation. In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, volume 2, pages 878–885 Vol.2, Dec 2003.

[51] Ikutaro Okada, Koji Takahashi, Wenqiang Zhang, Xiaofu Zhang, Hongyu Yang, and Shigeru Fujimura. A genetic algorithm with local search using activity list characteristics for solving resource-constrained project scheduling problem with multiple modes. *IEEJ Transactions on Electrical and Electronic Engineering*, 9(2) :190–199, 2014.

[52] Ramon Alvarez-Valdes Olaguibel and JoseManuel Tamarit Goerlich. The project scheduling polyhedron : dimension, facets and lifting theorems. *European Journal of Operational Research*, 67(2) :204–220, 1993.

[53] Y. Ouazene, F. Yalaoui, H. Chehade, and A. Yalaoui. Workload balancing in identical parallel machine scheduling using a mathematical programming method. *International Journal of Computational Intelligence Systems*, 7 :58–67, 2014.

[54] E Pinson, Christian Prins, and F Rullier. Using tabu search for solving the resource-constrained project scheduling problem. In *Proceedings of the 4th international workshop on project management and scheduling, Leuven, Belgium*, pages 102–106, 1994.

[55] Christian Prins, Caroline Prodhon, and Roberto Wolfler Calvo. Two-phase method and lagrangian relaxation to solve the bi-objective set covering problem. *Annals of Operations Research*, 147(1) :23–41, 2006.

[56] A. Alan B. Pritsker, Lawrence J. Watters, and Philip M. Wolfe. Multiproject scheduling with limited resources : A zero-one programming approach. *Management Science*, 16(1) :93–108, 1969.

[57] Anthony Przybylski, Xavier Gandibleux, and Matthias Ehrgott. Two phase algorithms for the bi-objective assignment problem. *European Journal of Operational Research*, 185(2) :509 – 533, 2008.

[58] Anthony Przybylski, Xavier Gandibleux, and Matthias Ehrgott. A two phase method for multi-objective integer programming and its application to the assignment problem with three objectives. *Discrete Optimization*, 7(3) :149–165, 2010.

[59] Majid Sabzehparvar and S Mohammad Seyed-Hosseini. A mathematical model for the multi-mode resource-constrained project scheduling problem with mode dependent time lags. *The Journal of Supercomputing*, 44(3) :257–273, 2008.

[60] L. Ben Said, S. Bechikh, and K. Ghedira. The r-dominance : A new dominance relation for interactive evolutionary multicriteria decision making. *IEEE Transactions on Evolutionary Computation*, 14(5) :801–818, Oct 2010.

[61] H. Sato, H. Aguirre, and K. Tanaka. *Controlling Dominance Area of Solutions and Its Impact on the Performance of MOEAs*, pages 5–20. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[62] H. Sato, H. Aguirre, and K. Tanaka. *Self-Controlling Dominance Area of Solutions in Evolutionary Many-Objective Optimization*, pages 455–465. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[63] Jason R. Schott. Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Master's thesis, Massachusetts Institute of Technology, May 1995.

[64] Margarita Reyes Sierra and Carlos A. Coello Coello. *Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ϵ-Dominance*, pages 505–519. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[65] R. Slowinski. Chapter 7 - multiobjective project scheduling under multiple category resource constraints. In ROMAN Słowinski, , and JAN Weglarz, editors, *Advances in Project Scheduling*, Studies in Production and Engineering Economics, pages 151 – 167. Elsevier, Oxford, 1989.

[66] R Słowinski, Bogdan Soniewicki, and J Weglarz. Mps-decision support system for multiobjective project scheduling. 1991.

[67] Roman Słowinski. Multiobjective network scheduling with efficient use of renewable and nonrenewable resources. *European Journal of Operational Research*, 7(3) :265–273, 1981.

[68] Roman Słowinski, Bolesław Soniewicki, and Jan Weglarz. Dss for multiobjective project scheduling. *European Journal of Operational Research*, 79(2) :220–229, 1994.

[69] Balram Suman. Multiobjective simulated annealing-a metaheuristic technique for multiobjective optimization of a constrained problem. *Foundations of Computing and Decision Sciences*, 27(3) :171–191, 2002.

[70] Genichi Taguchi. *Introduction to quality engineering : designing quality into products and processes*. 1986.

[71] Ekunda L Ulungu and Jacques Teghem. The two-phases method : An efficient procedure to solve biobjective combinatorial optimization problems. *Foundations of Computing and Decision Sciences*, 20(2) :149–165, 1995.

[72] Vicente Valls, Francisco Ballestín, and Sacramento Quintanilla. A population-based approach to the resource-constrained project scheduling problem. *Annals of Operations Research*, 131(1) :305–324, 2004.

[73] Peter JM Van Laarhoven and Emile HL Aarts. Simulated annealing. In *Simulated Annealing : Theory and Applications*, pages 7–15. Springer, 1987.

[74] Vincent Van Peteghem and Mario Vanhoucke. A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, 201(2) :409–418, 2010.

[75] David A. van Veldhuizen and Gary B. Lamont. Multiobjective evolutionary algorithm test suites. In *Proceedings of the 1999 ACM Symposium on Applied Computing*, SAC '99, pages 351–357, New York, NY, USA, 1999. ACM.

[76] Mario Vanhoucke. A genetic algorithm for net present value maximization for resource constrained projects. In *EvoCOP*, pages 13–24. Springer, 2009.

[77] Sanderson C Vanucci, Eduardo G Carrano, Rafael Bicalho, and Ricardo HC Takahashi. A modified nsga-ii for the multiobjective multi-mode resource-constrained project scheduling problem. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–7. IEEE, 2012.

[78] Ana Viana and Jorge Pinho de Sousa. Using metaheuristics in multiobjective resource constrained project scheduling. *European Journal of Operational Research*, 120(2) :359–374, 2000.

[79] X. Wang, F. Dugardin, and F. Yalaoui. An exact method to solve a bi-objective resource constraint project scheduling problem. *IFAC-PapersOnLine*, 49(12) :1038 – 1043, 2016. 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016, Troyes, France, 28—30 June 2016.

[80] X. Wang, F. Yalaoui, and F. Dugardin. Enhanced genetic algorithms to solve a multi-objective resource constraint project scheduling problem. *Journal of Intelligent Systems (soumis)*.

[81] X. Wang, F. Yalaoui, and F. Dugardin. Non-dominated sorting genetic algorithms for a multi-objective resource constraint project scheduling problem. *Flexible Service and Manufacturing Journal (soumis)*.

[82] X. Wang, F. Yalaoui, and F. Dugardin. Genetic algorithms hybridized with the self controlling dominance to solve a multi-objective resource constraint project scheduling problem. In *2017 IEEE International Conference on Service Operations and Logistics, and Informatics, Bari, Italy*, 2017 (accepté).

[83] X. Wang, F. Yalaoui, and F. Dugardin. A mapping technique to improve solutions for a multi-objective resource constraint project scheduling problem. In *12th Metaheuristics International Conference, Barcelona, Spain*, 2017 (accepté).

[84] Reza Zamani. A competitive magnet-based genetic algorithm for solving the resource-constrained project scheduling problem. *European Journal of Operational Research*, 229(2) :552–559, 2013.

[85] Hong Zhang, Xiaodong Li, Heng Li, and Fulai Huang. Particle swarm optimization-based schemes for resource-constrained project scheduling. *Automation in Construction*, 14(3) :393–404, 2005.

[86] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. Performance assessment of multiobjective optimizers : an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2) :117–132, April 2003.

[87] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms : Empirical results. *Evol. Comput.*, 8(2) :173–195, June 2000.

Xixi WANG

Doctorat : Optimisation et Sûreté des Systèmes

Année 2017

Ordonnancement de projet avec contraintes de ressources et aide à la décision multi-objectif

Cette thèse porte sur la résolution multi-objectif du problème d'ordonnancement de projet avec contraintes de ressources. Après avoir dressé un état de l'art sur le problème, nous le résolvons dans un premier temps avec les approches exactes : la méthode à deux phases et la méthode de partitionnement parallèle. Face à un problème NP-difficile, les méthodes exactes ne permettent de résoudre que des instances de petites tailles. Par conséquent, les méthodes approchées sont mises en œuvre pour traiter les problèmes de plus grandes tailles. Les algorithmes génétiques sont d'abord adoptés pour résoudre notre problème. Au-delà des schémas de base, nous proposons d'améliorer les solutions par plusieurs hybridations. Une recherche locale avec la méthode de Mapping est appliquée pour une meilleure exploration de l'espace de recherche. Nous considérons ensuite un cas spécial où les décideurs souhaitent réduire le nombre de solutions afin de faciliter leur travail. Nous avons donc réalisé les pré-sélections vis-à-vis d'un ensemble de solutions de grande taille. Pour ce faire, plusieurs alternatives de dominance de Pareto sont intégrées. Ces règles de dominances sont implémentées dans les schémas des algorithmes génétiques classiques et hybridés avec des recherches locales. Les résultats montrent que les hybridations considérées permettent d'améliorer significativement les méthodes de base. Nos recherches dans le futur proche s'appuient sur la résolution des problèmes plus complexes et en relation avec les cas industriels au plus proches de la réalité.

Mots clés : décision multicritère - systèmes d'aide à la décision - ordonnancement (gestion) - algorithmes génétiques - allocation des ressources.

Resource Constraint Project Scheduling Problem and Multi-objective Decision-making

This thesis deals with the multi-objective Resource Constraint Project Scheduling Problem (RCPSP). After a specific literature review, we solve the problem with exact approaches in the first place: the Two Phases Method and the Parallel Partitioning Method. Due to the NP-hardness of the problem, the exact methods are only able to solve small instances. For this reason, we apply the approximated methods to deal with larger problems. The genetic algorithms are firstly adopted to solve large-scaled instances. Moreover, we propose to improve the basic scheme with several hybridizations. A local search with Mapping Method is applied for a better exploration of the solution space. Next, we consider a special case where the decision-makers wish to reduce the number of solutions. Thus, in this part of the thesis, we try to select the most interesting solutions among the whole non-dominated front. For this purpose, several dominance relationships are considered as alternatives of the Pareto dominance. These dominance rules are implemented in the basic genetic algorithm schemes, as well as those with local search. The results show that the considered hybridizations enhance highly the basic method results. Our future research will highlight more complex multi-objective RCPSP problems and the industrial application.

Keywords: multiple criteria decision making - decision support systems - production scheduling - genetic algorithms – resource allocation.