# Energie, coopération méta-heuristiques et logique floue pour l'optimisation difficile

Julien Autuori

Thèse
de doctorat
de l'UTT

**Julien AUTUORI**

# Energie, coopération méta-heuristiques et logique floue pour l'optimisation difficile

**Spécialité :**
Optimisation et Sûreté des Systèmes

2014TROY0036

Année 2014

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*

**Julien AUTUORI**

*le 5 décembre 2014*

# Energie, coopération métaheuristiques et logique floue pour l'optimisation difficile

## JURY

| | | |
|---|---|---|
| M. N. ESSOUNBOULI | PROFESSEUR DES UNIVERSITES | Président |
| M. X. DELORME | MAITRE ASSISTANT - HDR | Rapporteur |
| M. A. HAMZAOUI | PROFESSEUR DES UNIVERSITES | Directeur de thèse |
| M. F. HNAIEN | MAITRE DE CONFERENCES | Directeur de thèse |
| M. A. NAKIB | MAITRE DE CONFERENCES | Examinateur |
| Mme E. SAHIN | MAITRE DE CONFERENCES - HDR | Examinateur |
| M. Z. SARI | PROFESSEUR | Rapporteur |
| M. F. YALAOUI | PROFESSEUR DES UNIVERSITES | Examinateur |

# Remerciements

Ce travail a été effectué dans de bonnes conditions à l'Université de Technologie de Troyes avec l'École Doctorale et le Laboratoire d'Optimisation des Systèmes Industriels. Je souhaite remercier toutes les personnes qui m'ont entouré pendant ces longues années où j'ai réalisé ma thèse.

Mes remerciements s'adressent tout particulièrement à mes directeurs de thèse, M Faicel Hnaien et M Farouk Yalaoui, pour m'avoir encadré. Bien sûr j'exprime ma gratitude envers les rapporteurs de ma thèse, pour avoir évalué ce manuscrit et porté de l'intérêt au travail réalisé.

Je salue mes amis doctorants, post-doctorants et permanents pour leur soutien et leur bonne humeur, ce qui a permis à la thèse de se dérouler dans un ambiance agréable et plus particulièrement mes camarades de bureau : William Guerrero, Andres Bernate Lara et Oussama Masmoudi.

Je tiens à exprimer ma reconnaissance à mes parents Chantal et Laurent, mon frère Sylvain, toute ma famille et mes amis pour leur soutien lors de ce travail doctoral.

**Energie, coopération métaheuristiques et logique floue pour l'optimisation difficile**


**Résumé**

Au cours de cette thèse, l'exploration de l'espace de solutions par des métaheuristiques est abordée. Les métaheuristiques sont des méthodes d'optimisation utilisées pour résoudre des problèmes NP-difficile. Elles explorent aléatoirement l'espace de recherche pour trouver les meilleures solutions. Dans un premier temps, l'ensemble des solutions est modélisé par un espace unidimensionnel par une Méthode de Conversion de l'Espace de recherche (MCE). Des métriques sont proposées pour évaluer l'exploration de l'espace de recherche par une métaheuristique en identifiant les zones explorées et inexplorées. Ces métriques sont utilisées pour orienter l'exploration de l'espace de recherche d'une méthode d'optimisation. La convergence est améliorée en accentuant la recherche dans les zones explorées. Pour sortir des minimums locaux, l'exploration est diversifiée en la dirigeant vers les zones inexplorées. En associant l'exploration du voisinage, des solutions et ces métriques cartographiques, il est possible d'améliorer les performances des métaheuristiques. Plusieurs algorithmes mono-objectifs et multiobjectifs sont implémentés en version classique, hybridés par la recherche locale et par la MCE. Le problème NP-difficile du *Flexible Job Shop Problem* (FJSP) est utilisé comme référence. Les expérimentations avec les algorithmes hybridés montrent une amélioration des performances.



**Mots clefs** : Optimisation combinatoire, Algorithmes d'approximation, Ordonnancement.


**Energy, cooperation metaheuristics and fuzzy logic for hard optimization**


**Abstract**

In this thesis, the solution space exploration by the metaheuristic is developed. The metaheuristics optimization methods are used to solve NP-hard problems. They explore randomly the search space to look for the best solutions. In a first step, the solution set is modeled by a one-dimensional space by a *Mapping* Method (MaM). Metrics are proposed to evaluate the search space exploration by a metaheuristic, identifying the explored and unexplored zones. These metrics are used to guide the search space exploration of an optimization method. The convergence is improved by emphasizing the research in the zones explored. To get out local minima, the exploration is diversified by pointing it towards the unexplored zones. Combining the neighbour discovery of the solutions and these mapping metrics, it is possible to improve the performance of metaheuristics. Several single-objective and multi-objective algorithms are implemented in the classic version, hybridized with local search and MaM. The NP-hard problem of *Flexible Job Shop Problem* (FJSP) is used as reference. The experimentations with hybridized algorithms show performance improved.



**Keywords** : Combinatorial optimization, Approximation algorithms, Production scheduling.

# Table des matières

# Introduction générale

La recherche opérationnelle étudie l'ensemble des problèmes de décision pour utiliser au mieux les ressources disponibles. Depuis 1940, lors de la création de la première équipe de recherche opérationnelle pour implanter optimalement des radars de l'armée anglaise, des méthodes toujours plus performantes se sont développées améliorant les résultats. Les problèmes traités par la recherche opérationnelle sont répartis sur deux niveaux : le stratégique et l'opérationnel. Les problèmes stratégiques regroupent en partie les choix d'investissements, d'implantation ou encore de dimensionnement. Les problèmes opérationnels sont variés : ordonnancement, gestion des stocks, affectation, prévisions, etc... La plupart de ces problèmes sont très présents dans la vie courante et économique. Par exemple, une manufacture doit trouver les meilleurs plans de production et place pour les machines dans l'atelier, minimiser la consommation de matières premières et d'énergie pour la fabrication de produits. Ou encore, une entreprise fournissant de l'électricité doit choisir où implanter les usines de production et le dimensionnement du réseau de distribution. Et, pour une famille en voyage qui doit choisir le chemin le plus court et le moins coûteux pour atteindre sa destination. Ces dernières années, les innovations technologiques ont permis de résoudre des problèmes d'optimisation de taille de plus en plus grande. La méthode la plus simpliste est d'énumérer toutes les solutions pour identifier la ou les meilleure(s). Mais, malgré la puissance de calcul des ordinateurs actuels, il faudrait plusieurs mois, voir plusieurs années pour énumérer toutes les solutions. Pour éviter l'énumération, des algorithmes, basés sur des modélisations mathématiques, ont été proposés pour résoudre ces problèmes sans passer par l'énumération. Un algorithme est une succession d'étapes simples nécessaires à la résolution d'un problème. Plusieurs algorithmes peuvent résoudre un même problème ; pour les évaluer on dénombre les étapes nécessaires pour chacun. La théorie de la complexité évalue le temps et les ressources utilisées par un algorithme pour résoudre un problème. Les problèmes sont répartis dans les classes de complexité, en fonction de la complexité du ou des meilleurs algorithme(s) connu(s) pour les rèsoudre. Les classes les plus utilisées sont : la classe P qui contient les problèmes résolus en un temps polynomial sur une machine de Turing déterministe. Les algorithmes qui les résolvent sont considérés comme efficaces. La classe NP est composée des problèmes qui peuvent être résolus en un temps polynomial par une machine de Turing nondéterministe. Un problème est difficile ou complet s'il est au moins aussi difficile à résoudre que tous les autres problèmes de sa classe. Un problème NP-difficile est dit NP-complet. Les problèmes du cycle hamiltonien, de clique maximale, de coloration de graphes et du sac à dos sont des

problèmes NP-difficiles.

Les problèmes de la classe P peuvent être résolus efficacement par des méthodes exactes qui trouvent la ou les solutions optimales sans avoir besoin de passer par l'énumération. Par contre, les problèmes de la classe NP sont résolus difficilement en un temps raisonnable par les méthodes exactes. Les métaheuristiques sont utilisées pour résoudre ces problèmes NP-Difficiles. Elles ont l'avantage de fournir de bonnes solutions en un temps acceptable et chacune peut être adaptée à plusieurs problèmes. Les deux caractéristiques qui déterminent leur exploration de l'espace de solutions sont la convergence et la diversification. La convergence donne le moyen à l'algorithme d'obtenir rapidement de bonnes solutions. La diversification permet à l'algorithme d'éviter les minimums locaux. L'influence de chacune déterminera l'efficacité de la métaheuristique. Ces deux caractéristiques sont définies par des opérateurs statiques. Par exemple, pour les Algorithmes Évolutionnaires, en anglais *Evolutionary Algorithms*, (EA) (Merkle, 1997), la convergence est assurée par l'opérateur de croisement et la diversification par celui de la mutation. Dans le cas de la Recherche Tabou, en anglais *Tabu Search*, (TS) (Glover, 1986), la convergence et la diversification sont assurées par la liste taboue. Pour l'Algorithme à Colonie de Fourmis, en anglais *Ant Colony Optimization*, (ACO) (Colorni et al., 1992), le taux dépôt et d'évaporation des phéromones assure la convergence et la diversification. Avec l'Optimisation par Essaims Particulaires, en anglais *Particiles Swarm Optimization* (PSO) (Kennedy and Eberhart, 1995), les poids de la meilleure solution explorée par la population et celle par la particule assurent respectivement la convergence et la diversification. Ces opérateurs et leurs paramétrages déterminent la qualité de l'exploration de l'espace de solutions d'une métaheuristique. Pour cela, il est important de bien choisir la métaheuristique à appliquer en fonction du problème et des objectifs à optimiser.

L'hybridation, c'est-à-dire l'ajout d'un module qui optimise l'exploration de l'espace de solutions, permet d'améliorer les performances des métaheuristiques. Plusieurs hybridations existent tels que : un Contrôleur à Logique Floue, en anglais *Fuzzy Logic Controler*, (FLC) la co-évolution, la Recherche Locale, en anglais *Local Search* (LS) et la collaboration entre plusieurs métaheuristiques. Le CLF, (Lau et al., 2009), évalue périodiquement la convergence de la population courante à l'aide de métriques. Périodiquement, en fonction l'évaluation des métriques, les paramétrages des opérateurs seront modifiés pour obtenir de meilleure convergence et diversification. Dans un algorithme co-évolutionnaire plusieurs populations différentes explorent chacune une partie de l'espace de solutions. Les algorithmes co-évolutionnaires sont divisés en deux catégories : les compétitifs et les coopératifs. Dans la première catégorie (Paredis, 1998), la compétition entre les populations permet de progresser dans la recherche de bonnes solutions. Pour la seconde catégorie (Potter and De Jong, 1994), l'échange d'information entre les espèces améliore les individus au fil des itérations. Une LS explore itérativement le voisinage d'une solution pour l'améliorer. Cette hybridation accélère la convergence de l'algorithme. Certaines métaheuristiques sont utilisées ensemble pour résoudre un problème donné. L'algorithme résultant profite des avantages de chacune des méthodes le composant.

Malgré cela, l'exploration de l'espace de solutions par les métaheuristiques et leurs hybridations reste peu étudiée. L'exploration des métaheuristiques est équilibrée entre convergence et diversification. Mais cet équilibre est déterminé uniquement en fonction des notes obtenues par les solutions explorées lors de la simulation. Ainsi, un EA ne garde que les meilleures solutions pour la prochaine itération, de même que la mémoire de l'ACO n'est mise à jour qu'avec les meilleures solutions et les particules d'un PSO dont le déplacement n'est influencé que par les meilleures solutions globale et locale. Cela va de même pour les hybridations tel que le FLC règle dynamiquement les paramètres en fonction de l'évolution de la métaheuristique dans l'espace des objectifs. À partir d'une solution, la LS cherche une meilleure solution selon la fonction objectif.

## 0.1 Problématique

La thèse est financée par la communauté d'agglomération du grand Troyes dans le cadre d'un projet consistant a étudier l'installation et la gestion d'un réseau électrique multi-source. Le grand Troyes prévoyant à terme la diminution de la part de l'électricité produite par les centrales nucléaires envisage le développement d'autres moyens de production, notamment les énergies renouvelables. Pour suppléer le réseau principal, l'agglomération de Troyes souhaite mettre en place plusieurs sources d'énergie. Le principe du réseau multi-source est de raccorder des sources d'énergie différentes tout en maintenant la qualité de l'électricité pour le consommateur.

Le projet est réalisé par une collaboration entre l'Université de Reims Champagne-Ardenne (URCA) et l'Université de Technologie de Troyes (UTT). Le Centre de Recherche en Sciences et Techniques de l'Information et de Communication (CReSTIC) de l'URCA apporte son expérience dans les domaines de l'énergie et de l'automatisme. Le CReSTIC fournit les modèles mathématiques pour le dimensionnent et l'exploitation du réseau électrique multisource, ainsi que les objectifs à optimiser. À l'UTT, le Laboratoire en Optimisation des systèmes Industriels (LOSI) a un savoir faire en recherche opérationnelle. Le LOSI propose des méthodes d'optimisation pour résoudre les problèmes modélisés par le CReSTIC.

Donc, notre problème initial est le dimensionnement et l'exploitation d'un réseau électrique où la production est décentralisée. La production distribuée est la production d'énergie électrique à l'aide d'installations de petite capacité raccordées à des niveaux de tension peu élevée au réseau électrique principal. Les avantages de la production décentralisée permet d'alimenter des sites éloignés du réseau principal, la valorisation des sources d'énergie fatale ou dérivée et l'autonomie du système local en cas d'incident sur le réseau. Par contre des inconvénients existent pour lors de son utilisation une faible participation au réglage de la fréquence sur le réseau électrique, une faible capacité a participer à la reconstitution de la qualité du signal électrique et pour certaines sources (éolien et solaire) beaucoup d'intermittence. La figure 1 présente un exemple d'un réseau de production distribué.

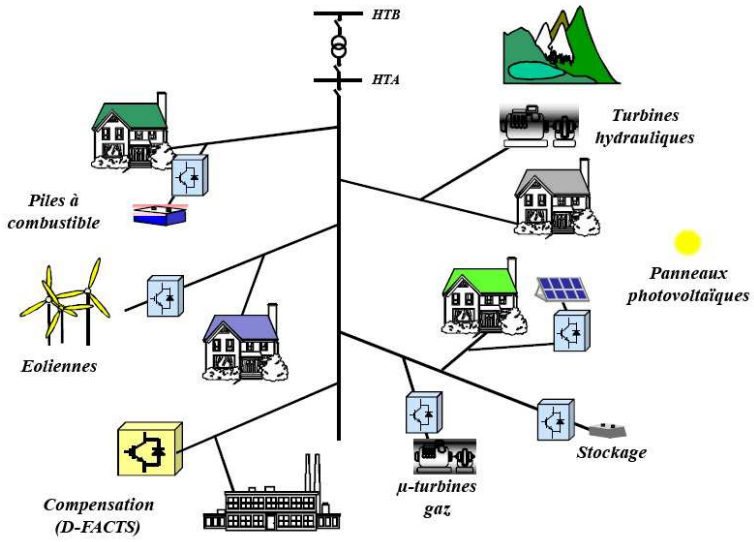


FIGURE 1 – Réseau électrique multisource

Voici une liste non-exhaustive, des sources d'énergie :

— La pile à combustible où la fabrication d'électricité se fait grâce à l'oxydation sur une électrode d'un combustible réducteur couplé à la réduction sur l'autre électrode d'un oxydant.
— Les éoliennes qui transforme l'énergie cinétique du vent, en énergie mécanique, puis en énergie électrique. L'énergie éolienne est renouvelable.
— L'énergie de la biomasse qui désigne l'ensemble des matières organiques d'origine végétale, animale ou fongique. Sans surexploitation de la ressource, elle est considérée comme renouvelable.
— Les panneaux photovoltaïques qui récupère le rayonnement solaire pour le convertir en énergie électrique. L'énergie solaire est renouvelable.
— La micro-cogénération qui est système de cogénération de très petite puissance électrique.
— La petite hydraulique qui consiste en une petite centrale électrique utilisant l'énergie hydraulique pour produire de l'électricité à petite échelle.
— Les turbines à gaz qui peuvent apporter leur appoint lors de pic de demande, mais sont rendement est faible.

Pour gérer le flux d'électricité des moyens de stockage sont nécessaires sur le réseau (batterie d'accumulateur et/ou condensateur). Des compensateurs (système de transmission flexible en courant alternatif) pour contrôler la tension, augmenter les capacités de transit, ou assurer la stabilité dynamique du réseau. Les objectifs a remplir lors de la gestion d'un tel réseau peuvent être multiple, en dehors du principal qui est de fournir de l'électricité (Par exemple : éviter de trop utiliser les moyens de stockage et savoir quand utiliser une source d'énergie spécifique).

Le problème de gestion d'un réseau de production décentralisée est certainement NP-Difficile et de grande taille. Pour résoudre ce problème, les méta-heuristiques sont les plus adaptées, mais

il est impossible de connaître la qualité des résultats à l'avance. L'étude de l'espace de recherche aide au choix de la méthode et les réglages qui donnent les meilleurs résultats. Notre approche consiste à faire une approximation de l'espace de recherche du problème NP-Difficile. Une fois cette approximation obtenue la topologie de cette carte est étudiée. Les différentes zones de la carte peuvent ÃŁtre identifiées. Pour chaque zone, la méthode la plus efficace effectue la recherche dans la zone concernée. Ceci implique de déterminer les différents types de topologie et de faire un inventaire de chacune des méthodes les plus efficaces pour les explorées.

Cette thèse présente une étude de l'exploration de l'espace de solutions par les métaheuristiques. Le principe étant d'évaluer l'exploration et ensuite de pouvoir orienter efficacement la recherche de la solution. Dans un premier temps, une modélisation efficace pour l'espace de solutions des divers problèmes d'optimisation combinatoire. L'une des formes les plus simples de représentation est un espace uni-dimensionnel, où chaque solution est représentée par un et un seul numéro. Pour obtenir cette carte unidimensionnelle, une Méthode de Conversion de l'Espace de recherche (MCE) est proposée. Elle utilise la conversion binaire de la représentation d'une solution, la distance de Hamming et une Fonction de Conversion Cartographique (FCC) bijective. Les solutions sont organisées par rapport à une solution de référence, les solutions avec beaucoup de différences avec la référence sont sur la carte trés éloignées de l'origine. La distance de Hamming est utilisée pour déterminer le nombre de différences par rapport à la référence. Malgré le fait que plusieurs solutions peuvent avoir la même distance de Hamming, la FCC attribue un numéro différent à chacune (Figure 2).

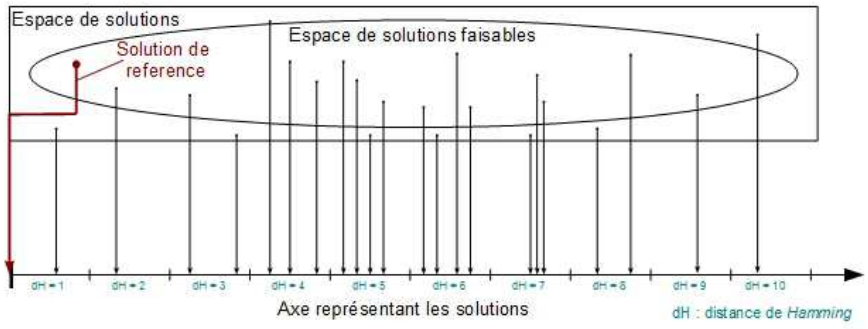


FIGURE 2 – Conversion de l'ensemble des solutions en un espace uni-dimensionnel

Cette modélisation donne une classification originale des solutions en fonction de leurs caractéristiques à la différence des métaheuristiques qui n'explorent l'espace de solutions qu'en fonction du ou des objectifs. Pour guider l'exploration des métaheuristiques, la carte unidimensionnelle est utilisée. Le découpage de la carte en zones permet de localiser plus facilement les solutions. La distance de Hamming peut être utilisée, chaque zone contient les solutions ayant le même nombre de différences. L'exploration de la carte par une métaheuristique est évaluée en qualifiant l'exploration de chaque zone. En dénombrant les solutions dans chaque zone, les zones explorées et inexplorées sont identifiées. Pour différencier les zones explorées, la qualité et le nombre des solutions explo-

rées contenues par chaque zone sont utilisés. Pour exploiter cette évaluation, plusieurs opérations existent telles que : la création de solutions dans les zones inexplorées et/ou l'application d'un traitement particulier aux solutions des zones explorées. L'objectif en orientant une métaheuristique est d'améliorer sa convergence et sa diversité. Par exemple, la création de solutions dans les zones inexplorées enrichit la diversité de la population. La création de solutions dans les zones inexplorées peut se faire en générant un nombre aléatoire et le convertir en sa solution correspondante ou en explorant le voisinage de solutions déjà explorées. Et pour améliorer la convergence, des zones explorées sont sélectionnées en fonction de la qualité de leurs solutions explorées. Les solutions de ces zones seront traitées pour trouver de meilleures solutions. La LS est utilisée pour exploiter les solutions dans les zones explorées. L'application de la LS est décidée en fonction de la qualité de l'exploration.

## 0.2 Contributions

Dans cette thèse, une modélisation originale de l'espace de solutions est proposée. Les solutions sont évaluées en fonction de leurs caractéristiques en plus de l'évaluation du ou des objectif(s). Cette modélisation est utilisée pour faire une évaluation de l'exploration de l'espace de solutions par une métaheuristique. L'évaluation est ensuite utilisée pour orienter la métaheuristique en accentuant la recherche dans des zones prometteuses explorées et en s'intéressant aux zones inexplorées en y créant des solutions.

— Le chapitre "Méthode de Conversion de l'Espace de recherche" (MCE), présente une procédure de conversion de l'ensemble des solutions d'un problème combinatoire en un espace uni-dimensionnel. La bijectivité de la MCE est démontrée. Des métriques d'évaluation de la convergence et diversification cartographique sont proposées pour la carte découpée en zones. Les zones sont ensuite sélectionnées en fonction de la qualité de leurs solutions explorées. Des algorithmes permettant de créer ou de modifier des solutions dans les zones sélectionnées sont proposés.

— Dans le chapitre "Application de la MCE au *Flexible Job Shop Problem*", le FJSP, est utilisé pour expérimenter la MCE. Le FJSP est un problème d'ordonnancement modélisant les ateliers produisant de petites séries de lots. Il est NP-difficile (Garey et al., 1976). Dans la première partie, le FJSP avec l'objectif de minimiser le *makespan* est résolu par un Algorithme Génétique, en anglais *Genetic Algorithm* (GA) de la littérature et comparé avec son hybridation avec la MCE. La seconde partie aborde le cas du FJSP multiobjectif. Les objectifs traités sont la minimisation du *makespan* et la production juste à temps des lots. Les algorithmes utilisés et hybridés par la MCE sont le *Non-dominated Sorting Genetic Algorithm - II* (NSGA-II) (Deb et al., 2002), le *Strengh Pareto Evolutionary Algorithm 2* (SPEA2) (Zitzler et al., 2001), le *MultiObjective Artificial Immune Système* (MOAIS), *MultiObjective Ant Colony Optimzation* (MOACO), le *MultiObjective Particle Swarm Optimzation* (MOPSO) et

le *MultiObjective Artificial Bee Colony* (MOABC). L'exploration de l'espace de solutions de chaque métaheuristiques est étudiée et plusieurs politiques d'amélioration sont présentées.

## 0.3 Organisation

Le premier chapitre contient un état de l'art sur les diverses méthodes d'optimisations. Les deux premières parties présentent les deux familles de méthodes : les méthodes exactes et les métaheuristiques. Les métaheuristiques sont réparties en deux sous-parties : celles qui utilisent qu'un seul individu et celles qui ont une population pour explorer l'espace de solutions. Une dernière partie présente les hybridations des métaheuristiques les plus courantes.

Le deuxième chapitre présente les étapes de la MCE qui utilise la conversion binaire, la distance de *Hamming* et la FCC. La bijectivité de la FCC est prouvée. Les métriques cartographiques évaluent l'exploration de la carte divisée en zones. Des algorithmes sont proposés pour utiliser cette évaluation. La deuxième partie du chapitre contient un état de l'art sur le FJSP, sa formulation, les objectifs les plus communs, les *benchmark* de référence et des générateurs d'instances. La MCE a été utilisée avec un Algorithme Génétique (AG) pour le FJSP avec un objectif, la minimisation du *makespan*.

Dans le troisième chapitre, après un état de l'art sur le FJSP multiobjectif, le FJSP avec les objectifs de minimisation du *makespan* et la production des lots juste à temps a été utilisé pour comparer les collaborations entre la MCE et les métaheuristiques multi-objectifs NSGAII, SPEA2, MOAIS, MOACO, MOPSO et MOABC. Les opérateurs utilisés par ses méthodes sont décrit. Les résultats des méthodes d'optimisation sont évalués par des métriques classiques et cartographiques.

# Chapitre 1

# État de l'art sur les méthodes d'optimisation

## 1.1 Introduction

Les méthodes d'optimisation ont fait l'œuvre de nombreux travaux, proposant de nombreuses variantes. Pour les problèmes de la classe P, il existe des algorithmes spécifiques pour chacun qui permettent d'obtenir la solution optimale en un temps raisonnable et ils ne sont pas abordés dans cette thèse. Ce chapitre présente les principales catégories de méthodes d'optimisation pour résoudre les problèmes de la classe NP. Dans un premier temps, quelques définitions sur les problèmes d'optimisation sont présentées, notamment pour distinguer l'optimisation mono-objectif et multiobjectif. Ensuite, les méthodes d'optimisation exactes, les méthodes approchées avec population et celles sans population sont présentées. L'état de l'art listera les principales méthodes d'optimisation en abordant quelques versions pour résoudre les problèmes mono-objectifs et leur adaptation pour l'optimisation multiobjectif. Enfin, des techniques d'hybridation qui permettent d'améliorer les performances des méthodes d'optimisation sont aussi présentées.

## 1.2 Problèmes d'optimisation

Un problème d'optimisation consiste à minimiser ou maximiser une fonction objectif. Cette fonction comporte des variables de décisions soumises à des contraintes. Les contraintes déterminent la complexité du problème (classe P ou NP). Des modélisations mathématiques sont utilisées pour les représenter. Par la suite, des méthodes d'optimisation utilisent ces modélisations pour résoudre les problèmes.

Un problème d'optimisation se modélise par une fonction objectif et des contraintes.

La fonction objectif $f$ est définie par :

$$f : A \to \mathbb{R}$$

avec l'ensemble de solutions $S$ défini par des contraintes.

$$S \subset \mathbb{R}^n$$

où $n$ est le nombre de variables.

Le domaine de $A$ est appelé espace de solutions, les éléments contenus dans $S$ sont des solutions réalisables.

Une solution réalisable $u_i$ qui minimise $f$ est appelée solution optimale.

$$f(u_i) \leq f(u_j) \quad \forall s_j \in S$$

Un exemple simple de problème d'optimisation est la tournée de véhicules. Un bus effectue le ramassage scolaire. Il doit respecter des contraintes sur les horaires. Son objectif est de faire le trajet le plus court pour minimiser les coûts.

De nombreux problèmes nécessitent l'optimisation d'objectifs souvent contradictoires. Par exemple lors d'un voyage, les deux objectifs à remplir sont la minimisation du temps et du coût. Le fait de minimiser le coût peut entraîner l'augmentation du temps de voyage.

La fonction objectif d'un problème multiobjectif (Van Veldhuizen, 1999) (Coello Coello and Lamont, 2004) (Coello Coello, 2002) est un vecteur de fonctions mono-objectifs défini par :

$$F(x) = [f_1(x), f_2(x), ..., f_k(x)]$$

chaque fonction $f_i(x)$ peut-être à maximiser ou à minimiser.

où $x$ est le vecteur représentant les $n$ variables de décision :

$$x = [x_1, x_2, ..., x_n]$$

Comme dans le cas du mono-objectif l'ensemble des solutions réalisables $S$ est défini par des contraintes.

L'ensemble de tous les n-tuples de nombres réels dans $\mathbb{R}^n$ est appelé le n-espace Euclidien. Deux espaces euclidiens sont considérés dans les problèmes multiobjectifs :

— Un espace à $n$ dimensions où chaque axe est celui d'une des variables composant le vecteur $x$.
— Un autre à $k$ dimensions pour les fonctions objectifs constituant la fonction objectif $F(x)$.

Tous les points dans l'espace des variables représentent une solution qui renvoie à un second point dans l'espace des objectifs. Ce deuxième point détermine la qualité de la solution donnée par la fonction objectif.

Pour un problème mono-objectif, la comparaison de deux solutions est simple. Mais lorsque plusieurs critères sont pris en compte, il est plus difficile de définir la qualité d'une solution par rapport à une autre. Pour comparer deux solutions, il faut définir une notion de dominance. La dominance de Pareto est certainement la plus utilisée des règles de dominance. Les règles de la dominance de Pareto sont définies comme suit :

**Définition 1.** Un vecteur $u = (u_1, ..., u_k)$ domine un autre vecteur $v = (v_1, ..., v_k)$ ($u \preceq v$) si et seulement si $u$ est partiellement meilleur que v, c'est à dire dans le cas d'une minimisation, $\forall i \in \{1, ..., k\}$, $u_i \leq v_i \wedge \exists i \in \{1, ..., k\} : u_i < v_i$.

**Définition 2.** Un point de $x^* \in S$ est optimal de Pareto au sens faible s'il n'y a pas de $x \in S$ tel que $f_i < f_i(x^*)$, pour $i = 1, ..., k$.

**Définition 3.** Un point de $x^* \in S$ est optimal de Pareto au sens stricte s'il n'y a pas de $x \in S$, $x \neq x^*$ tel que $f_i \leq f_i(x^*)$, pour $i = 1, ..., k$.

**Définition 4.** Deux solutions $u$ et $v$ sont dites non-dominées ou incomparables ($u \sim v$) si $\exists i \leq m, f_i(v) < f_i(u)$ et $\exists j \leq m, f_j(v) > f_j(u)$ ou si $\forall i \leq m, f_i(v) = f_i(u)$.

Avec ces définitions trois types de zones sont identifiées autour d'une solution $u$ : la zone de dominance qui contient toutes les solutions dominants $u$, la zone de préférence avec toutes les solutions dominées par $u$ et la zone d'indifférence regroupant toutes les solutions non-dominées par $u$. Pour trouver les meilleures solutions, les définitions suivantes sont nécessaires.

**Définition 5.** Pour un problème multiobjectif F(x), l'ensemble de Pareto optimal $\mathcal{P}^*$ est défini par :

$$\mathcal{P}^* := \{x \in S | \neg \exists x^{'} \in SF(x^{'}) \preceq F(x)\}$$

**Définition 6.** Pour un problème multiobjectif F(x) et un ensemble de Pareto optimal $\mathcal{P}^*$, le front de Pareto est défini par :

$$\mathcal{PF}^* := \{u = F(x) | x \in \mathcal{P}^*\}$$

Les algorithmes pour l'optimisation mono-objectif sont adaptés pour l'optimisation multiobjectif en utilisant la dominance de Pareto. Les méthodes mono-objectif retournent les solutions avec la meilleure note obtenue, tandis que celles pour le multiobjectif renvoient le front de Pareto optimal trouvé. En fonction du nombre d'objectifs et des contraintes, les méthodes ont des performances diverses pour un même problème. Dans cette thèse, diverses familles de méthodes d'optimisation sont étudiées en mono et multiobjectif : l'algorithme génétique, le système immunitaire artificiel, l'optimisation par colonie de fourmis, l'optimisation par essaims particulaires et l'algorithme à

colonies d'abeilles. L'objectif étant par la suite de pouvoir les améliorer par une exploration originale de l'espace de solutions. Dans les sections suivantes, les méthodes d'optimisations connues sont présentées tandis que celles étudiées sont décrites davantage.

## 1.3 Méthodes exactes

Les méthodes exactes donnent la ou les solution(s) optimale(s) sans énumérer toutes les solutions réalisables possibles. Le gain de temps est donc substantiel. De nombreuses méthodes exactes sont développées, comme la programmation par contraintes, la programmation dynamique, la séparation et évaluation et la programmation linéaire. La plupart des algorithmes pour l'optimisation mono-objectif sont adaptés pour les problèmes multiobjectifs, mais certains sont spécifiques à l'optimisation multiobjectif, par exemple les méthodes à deux phases et à partitions parallèles.

### 1.3.1 Programmation par contraintes

La programmation par contraintes est une technique de résolution des problèmes combinatoires complexes apparue à la fin des années 1980. Dans l'une des premières références à la programmation par contraintes (Montanari, 1974), plusieurs manipulations de contraintes sont comparées pour des problèmes de reconnaissance d'images et de production syntaxique du langage. Deux représentations de chaque problème sont proposées l'une binaire et une autre non-binaire. Un exemple montre un tableau de n contraintes qui ne peut pas être transformé en contraintes binaires avec n variables. Donc les contraintes binaires et non-binaires ne sont pas équivalentes, les contraintes non-binaires semblent être plus efficaces que les binaires. Mais ceci, n'est vrai que si le nombre de variables est limité dans le problème transformé.

La programmation par contraintes est utilisée pour résoudre de nombreux problèmes. Par exemple, le planning des infirmières d'un hôpital (Weil et al., 1995), l'objectif est de spécifier les jours et les horaires de service des infirmières. Ce problème prend en compte plusieurs contraintes essentiellement issues du code du travail.

L'une des études les plus récentes présente le *Dynamic Constraint Satisfaction Problems* (DynCSPs) (Climent et al., 2012). Le DynCSPs prend en compte les changements éventuels intervenant durant l'exécution du programme. Le problème est de traiter le modèle *Weighted Constraint Satisfaction Problem* (WSCP). Une solution robuste pour le DynCSPs est une bonne solution pour le WCSP.

### 1.3.2 Programmation dynamique

La programmation dynamique (*Dynamic programming*) (Bellman and Kalaba, 1965) s'applique à des problèmes d'optimisation dont la fonction objectif consiste à optimiser des sommes de fonctions

monotones croissantes sous contraintes de croissance. Le principe est simple : le problème peut être décomposé en sous-problèmes et la solution optimale du problème est la somme des solutions optimales des sous-problèmes. Les sous-problèmes sont calculés des plus petits aux plus grands.

La programmation dynamique est utilisable pour résoudre des problèmes multiobjectifs. Pour la résolution des problèmes multiobjectifs par la programmation dynamique, un exemple (Liao and Li, 2002) propose une structure générale qui est séparable. Dans la fonction objectif, des coefficients de pondération sont utilisés pour chaque objectif, ils sont variables durant la simulation. La facilité d'utilisation pour les problèmes multiobjectifs est l'avantage de cette méthode de programmation dynamique.

Un algorithme de programmation dynamique se ramène à un algorithme de recherche de plus court chemin (Martelli, 1976). Le problème étudié consiste à détecter les contours d'une image floue. L'auteur a prouvé que ce problème peut être représenté par un problème du plus court chemin. Il est donc plus avantageux d'utiliser un algorithme pour trouver le plus court chemin que la programmation dynamique. Or, l'un des algorithmes les plus efficaces pour trouver le plus court chemin est l'algorithme A* (Hart et al., 1968), il est aussi utilisé en intelligence artificielle. Cet Algorithme est basé sur une recherche utilisant une théorie mathématique de la recherche dans un graphe optimal. L'algorithme $A^*$ n'explore pas plus de nœuds que tous les autres algorithmes de recherche de plus court chemin, ce qui en fait l'algorithme le plus efficace pour trouver le plus court chemin.

Une adaptation de l'algorithme $A^*$ pour les problèmes multiobjectifs combinatoires ($MOA^*$) identifie les chemins non-dominés à partir d'un nœud vers un ensemble donné de nœuds à atteindre (Stewart and White III, 1991). Si le MOA* est utilisé pour un problème monoobjectif il fonctionne comme le A*, car il utilise les mêmes principes et théories mathématiques.

### 1.3.3 Séparation et évaluation

L'algorithme par séparation et évaluation (Land and Doig, 1960), en anglais *Branch and Bound* (B&B), est généralement utilisé pour résoudre des problèmes d'optimisation combinatoire. Les solutions peuvent être toutes énumérées, mais en utilisant les propriétés du problème l'énumération est évitée. Donc, logiquement, un bon B&B n'explore que les solutions potentiellement intéressantes.

Dans la phase de séparation, le problème est divisé en sous-problèmes. La réunification des espaces de solutions de chacun des sous-problèmes correspond à la totalité des solutions possibles pour le problème initial. La séparation peut être aussi appliquée récursivement à chaque sous-problème. Les meilleures solutions de chaque sous-problème sont ensuite comparées pour trouver la meilleure solution.

Lors de la séparation, un arbre de recherche est fabriqué. L'évaluation de chaque nœud consiste à calculer une borne inférieure, pour le cas d'une minimisation, le sous-ensemble de solutions auquel

il est associé. Si la borne inférieure est supérieure aux solutions déjà trouvées, alors il existe donc une preuve mathématique pour arrêter l'exploration du sous-espace.

Le *MultiObjective depth-First Branch and Bound for optimization task* (MO-B&B) (Rollon and Larros, 2009) est une adaptation du B&B pour les problèmes d'optimisation multiobjectif. En premier lieu, le MO-B&B cherche la profondeur de l'arbre défini par le problème. Durant la recherche, le MO-B&B utilise une archive contenant les solutions non-dominées. Quand une nouvelle solution est trouvée, une mise à jour est faite. Comme dans le B&B à chaque évaluation, une borne inférieure est calculée ; si cette borne inférieure est dominée par les solutions dans l'archive, alors, la recherche dans ce sous-espace associé peut s'arrêter.

### 1.3.4 Programmation linéaire

La programmation linéaire (Dantzig and Cottle, 2003) s'intéresse à la maximisation ou à la minimisation d'une fonction objectif linéaire définie sur un ensemble fini de variables qui vérifient un nombre fini de demi-espaces : un polyèdre convexe. Les contraintes définies peuvent être écrites par des fonctions linéaires.

Un problème de programmation linaire peut s'écrire comme suit :

$$\begin{aligned} min \quad & c^T x \\ s.t. \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{1.1}$$

Où $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ et la matrice de contrainte $A$ de dimension $m \times n$.

L'algorithme du simplex est la première méthode de programmation linéaire (Dantzig and Cottle, 2003), il minimise ou maximise une fonction objectif dans un ensemble défini d'inégalités.

La deuxième méthode, celle des points intérieurs (Dantzig and Thapa, 2003), contrairement au simplex, traverse l'espace de solutions réalisables pour atteindre la solution optimale.

La méthode des ellipsoïdes (Shor, 1977) concurrence l'algorithme du simplex, car elle donne la solution en un temps polynomial. Cette méthode explore le polyèdre convexe en le réduisant pour trouver rapidement l'optimum global.

La méthode des points intérieurs (Vanderbei et al., 1986) est adaptée pour l'optimisation multiobjectif (Abel and Korhonen, 1996). À chaque itération, un opérateur de décision choisit le poids à accorder à chaque objectif et une fonction scalaire permet de trouver la ou les solution(s) non-dominée(s) pour ce vecteur. L'opérateur de décision permet de choisir à chaque itération une nouvelle zone à explorer pour trouver de nouvelles solutions non-dominées.

### 1.3.5 Méthodes exactes pour l'optimisation multiobjectif

La méthode à deux phases (Ulungu and Teghem, 1995), en anglais *Two Phases Method* (TPM), ne résout que les problèmes d'optimisation combinatoires et convexes bi-objectifs. Elle est, comme son nom l'indique, composée de deux étapes. La première étape consiste à trouver les deux points extrêmes du front Pareto et de trouver les solutions supportées. Lors de la deuxième étape, les solutions non-supportées sont recherchées entre les solutions supportées. Les solutions non-supportées se trouvent dans les triangles rectangles formés par deux solutions supportées adjacentes. Les algorithmes utilisés pour trouver les solutions extrêmes, les solutions supportées et les solutions non-supportées dépendent du problème à résoudre (Visée et al., 1998) (Przybylski et al., 2008).

La méthode à partitions parallèles (Lemesre et al., 2007), en anglais *Parallel Partitioning Method* (PPM), est dérivée de la TPM et résout des problèmes d'optimisation bi-objectifs. Elle est composée de trois étapes. La première consiste à trouver les solutions extrêmes du front de Pareto. Lors de la seconde, l'espace de solutions est découpé entre les deux bornes du front de Pareto optimal. Le découpage se fait sur la dimension d'un objectif. Ensuite pour chaque zone une solution optimale est calculée en utilisant la méthode $\epsilon - contrainte$ (Haimes et al., 1971). La troisième trouve les solutions dans les rectangles formés par deux solutions optimales adjacentes définies lors des étapes précédentes.

L'un des avantages de la PPM sur la TPM est qu'elle n'a pas besoin de rechercher toutes les solutions supportées, ce qui réduit le coût en calcul. Dans la PPM, le nombre de partitions est à déterminer par l'utilisateur ce qui fournit un contrôle sur le nombre de calculs réalisés. La PPM obtient aussi des sous-espaces de taille plus uniforme que la TPM, ce qui empêche des recherches dans des sous-espaces démesurément grands et donc minimise le nombre de calculs. Mais durant la deuxième phase de la PPM, les partitions ajoutent une contrainte sur un objectif, ce qui peut fausser le problème initial.

La méthode à partitions parallèles - K (Dhaenens et al., 2010), en anglais *K - Parallel Partitioning Method* (K-PPM), est une adaptation de la PPM pour les problèmes avec plus de deux objectifs. La première phase est similaire à celle de la TPM et de la PPM. Elle consiste à trouver les solutions extrêmes sur le front de Pareto. Ces solutions sont les sommets de l'ensemble qui contient toutes les solutions non-dominées. La seconde phase découpe uniformément l'espace de recherche en hyperplans de dimension K. Pour cela, un objectif est choisi et l'espace est partagé en boîtes de taille uniforme sur la dimension de ce dernier. La troisième phase recherche les solutions restantes dans les boites définies précédemment.

## 1.4 Méthodes approchées avec population

Les méthodes exactes donnent toujours les optimums globaux, mais pour les instances de grandes tailles, des problèmes NP-difficiles le temps d'exécution est beaucoup trop grand. Les méthodes

approchées réduisent le temps de calculs, les résultats sont de bonne qualité, mais les solutions ne sont pas optimales. Ces méthodes peuvent se diviser entre celles avec population et celle sans. Un algorithme avec une population explore plusieurs solutions en même temps, ce qui procure un avantage quand l'espace de recherche est composé de nombreux minimums locaux. Les individus peuvent se servir des informations des autres pour sortir des minimums locaux. Par contre, gérer une population peut s'avérer complexe. Par exemple, la population ne doit pas être concentrée dans la même zone de l'espace de recherche enlevant ainsi l'intérêt d'utiliser une population. Ceci est important pour la population initiale dont la qualité des solutions doit être aussi bonne pour donner un bon départ à la métaheuristique. La technique de communication entre les individus différencie une métaheuristique d'une autre car elle définit l'échange d'informations et donc l'exploration de l'espace de recherche. Les *Evolutionary Algorithms* (EA)s utilisent le croisement, c'est-à-dire le découpage des solutions en morceaux et leur ré-assemblage. Les *Ant Colony Optimization* (ACO)s gardent en mémoire les meilleures solutions pour en reconstruire à l'itération suivante. Pour le *Scatter Search* (SS), les nouvelles solutions sont construites entre deux déjà existantes. L'*Artificial Bee Colonie* (ABC) opère l'exploration du voisinage d'une solution à l'aide d'une autre. Lors d'une itération de l'*Artificial Immune System* (AIS), les meilleurs individus ont plus de chance d'être sélectionnés pour le clonage. L'expérience des individus est partagée pour améliorer l'exploration de l'espace de recherche dans le *Q-learning* (QL). Les EAs sont une famille très variée et populaire regroupant plusieurs catégories ; les plus connus sont les *Genetic Algorithms* (GA)s, mais il existe aussi les *Differential Evolution* (ED), la *Harmony Search* (HS) et la *Cross Entropy* (CE).

### 1.4.1 Algorithmes évolutionnaires

Les algorithmes évolutionnaires, en anglais *Evolutionary Algorithm* (EA), (Merkle, 1997) (Fogel, 1997) sont inspirés de l'évolution des espèces au fil des générations pour s'adapter aux modifications de leur environnement (voir algorithme 1.1). Cette évolution est basée sur quatre opérateurs : la sélection, le croisement, la mutation et l'évaluation. L'opérateur de sélection choisit les parents pour le croisement. Le croisement échange les informations entre individus pour en créer de nouveau, ceci permet à l'algorithme de converger. La mutation modifie des individus nouvellement crées par le croisement et permet ainsi à l'algorithme d'éviter les optimums locaux. L'évaluation donne une note à chaque individu et estime la diversité de la population.

**Algorithme 1.1** Schéma d'un algorithme évolutionnaire

**Paramètre** $n$ : Taille de la population
**Paramètre** $m$ : Nombre de couples
**Paramètre** $t_c$ : Le taux de croisement
**Paramètre** $t_m$ : Le taux de mutation
**Paramètre** $T$ : Critère d'arrêt
  $P_0 \leftarrow$ Initialisation($n$)
  $t \leftarrow 0$
  **Tant que** T n'est pas atteint **Faire**
    $C_t \leftarrow$ SélectionParents($m$,$P_t$)
    $E_t \leftarrow$ Croisement($t_c$,$C_t$)
    $E_t \leftarrow$ Mutation($t_m$,$E_t$)
    Évaluation($P_t$,$E_t$)
    $P_{t+1} \leftarrow$ SélectionMeilleurs($n$,$P_t$,$E_t$)
    $t \leftarrow t + 1$
  **Fin Tant que**
  **Retourne** Meilleure solution de $P_t$

Les algorithmes évolutionnaires ont plusieurs adaptations pour l'optimisation multiobjectif.

Le *MultiObjective Genetic Algorithm* (MOGA) (Fonseca and Fleming, 1993) utilise une technique d'évaluation qui consiste à compter pour chaque individu d'une génération le nombre d'individus qui le domine. C'est ainsi qu'est déterminé le rang d'une solution explorée.

Le *Nondominated Sorting Genetic Algorithm* (NSGA) (Yaochu et al., 2002) est basé sur plusieurs étapes de classification des individus. Pour maintenir la diversité, les individus de la population sont classés selon la valeur de leur évaluation. Les individus sont donc classifiés en fronts. Une sélection aléatoire proportionnelle à la valeur du front est effectuée pour choisir les parents. Les solutions avec les meilleures notes ont plus de chance d'être sélectionnées comme parents, pour permettre une meilleure convergence et en visitant des régions moins explorées.

Le *Non-dominated Sorting Genetic Algorithm-II* (NSGA-II) (Deb et al., 2002) est le plus populaire des algorithmes évolutionnaires multiobjectifs. Le principe est le même que le NSGA, mais la sélection des individus diffère (voir algorithme 1.2). Les parents et les enfants sont réunis dans la même population et sont classés par front de Pareto. À l'intérieur de chaque front de Pareto, le *crowding distance* est calculé pour chaque individu. Le principe du *crowding distance* est d'étirer le front de Pareto optimal. Pour cela, les solutions extrêmes du front de Pareto sont privilégiées dans la notation ainsi que les individus les plus éloignés de leur voisin. Pour le calcul du *crowding distance*, les notes initiales d'un front de Pareto sont initialisées à 0. Pour chaque objectif, les solutions sont triées par ordre croissant et la note de chaque solution est modifiée par l'équation 1.2. La note des solutions en première et dernière place reçoit la valeur infinie ($\infty$).

$$NCD_i = NCD_i + (NO^k_{i+1} - NO^k_{i-1})/(NO^k_{max} - NO^k_{min}) \quad (1.2)$$

Où $NCD_i$ note au *crowding distance* de la solution à la position $i$ dans le classement et $NO^k_i$ la note obtenue à l'objectif $k$ pour la solution $i$. $min$ et $max$ sont les indices respectifs de la première et de la dernière solution dans le classement. La nouvelle population est complétée en partant des individus du meilleur front de Pareto, jusqu'atteindre la limite de population. Si, en ajoutant les individus d'un front, la limite de population est atteinte, le choix des individus est déterminé par leur note obtenue par le *crowding distance*.

**Algorithme 1.2** Algorithme NSGA-II

**Paramètre** $n$ : Taille de la population
**Paramètre** $m$ : Nombre de couple
**Paramètre** Critère d'arrêt
1: $P_0 \leftarrow$ Initialisation($n$)
2: $t \leftarrow 0$
3: **Tant que** Critère d'arrêt n'est pas atteint **Faire**
4: $C_t \leftarrow$ SélectionParents($P_t$)
5: $E_t \leftarrow$ Croisement($C_t$)
6: $E_t \leftarrow$ Mutation($E_t$)
7: *Non-Dominated Sorting*($P_t$, $E_t$)
8: *Crowding Distance*($P_t$, $E_t$)
9: $P_{t+1} \leftarrow$ SélectionMeilleurs($P_t$, $E_t$)
10: $t \leftarrow t + 1$
11: **Fin Tant que**
12: **Retourne** Front de Pareto optimal dans $P_t$

Le *Niched Pareto Genetic Algorithm* (NPGA) (Horn et al., 1994) diffère des autres algorithmes génétiques sur la sélection des individus pour la reproduction. La sélection est basée sur la notion de la dominance de Pareto. Deux individus sont sélectionnés aléatoirement et l'individu dominant est choisi. Si les deux ont le même rang, alors ils sont départagés aléatoirement.

Le *Pareto Archived Evolution Strategy* (PAES) (Knowles and Corne, 1999) consiste en une stratégie d'évolution (1+1), c'est-à-dire qu'un parent engendre un enfant. Une archive contient toutes les solutions non-dominées explorées. Cette archive est utilisée comme référence à laquelle est comparée chaque individu ayant muté. Le PAES utilise aussi une procédure de *crowding* qui divise récursivement l'espace des objectifs, ce qui permet d'évaluer la diversité de la population. L'espace de solutions est divisé en fonction des objectifs pour déterminer le nombre d'individus dans chaque zone.

Le *Strength Pareto Evolutionary Algorithm* (SPEA) (Zitzler and Thiele, 1999) utilise plusieurs éléments des algorithmes évolutionnaires. Le SPEA utilise une archive contenant toutes les solutions

non-dominées déjà trouvés, à chaque itération l'archive est mise à jour. Le nombre d'individus dominés est calculé pour chaque individu dans l'archive. Cette évaluation permet d'identifier les individus du front de Pareto et d'estimer la diversité de la population. Ceci permet de sélectionner des individus bien répartis sur le front de Pareto.

Le *Strength Pareto Evolutionary Algorithm 2* (SPEA2) (Zitzler et al., 2001) est une amélioration du SPEA. Par rapport au SPEA, il utilise la stratégie du *fine-grained* pour calculer le nombre d'individus dominé par un autre (voir algorithme 1.3). La densité autour de chaque individu est estimée en identifiant le plus proche voisin. Un opérateur de troncature assure la sélection des solutions sur le front de Pareto lors de l'archivage.

Le *fine-grained* est une procédure qui se déroule en plusieurs étapes. Pour commencer, le nombre de solutions dominées $S(i)$ est calculé pour chaque solution.

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

Cette donnée permet de calculer pour chaque individu le nombre de solutions $R(i)$ qui le domine. Un individu non-dominé a une valeur de $R(i) = 0$.

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

Pour départager des individus ayant la même note $R(i)$, une troisième valeur $D(i)$ est introduite. Elle permet de privilégier les individus dans les zones peu denses de l'espace des objectifs. Pour cela $D(i)$ est déterminée par :

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

avec $\sigma_i^k$ la distance euclidienne dans l'espace des objectifs du $k^{me}$ plus proche voisin. Il est conseillé de prendre $k = \sqrt{N + \overline{N}}$.

Le résultat de la fonction d'évaluation est ensuite calculé en sommant $R(i)$ et $D(i)$.

$$F(i) = R(i) + D(i)$$

$F(i)$ est utilisé au cas où le nombre d'individus non-dominés est supérieur à la taille $\overline{N}$ de l'archive $A$. Dans ce cas, seuls les $\overline{N}$ premiers individus dans le classement par ordre croissant selon $F(i)$ sont sauvegardés dans $A$.

**Algorithme 1.3** Algorithme SPEA2

**Paramètre** $N$ : Taille de la population
**Paramètre** $\overline{N}$ : Taille de l'archive
**Paramètre** $T$ : Critère d'arrêt

1: $P_0 \leftarrow$ Initialisation($N$)
2: $A_0 \leftarrow \oslash$
3: $t \leftarrow 0$
4: **Tant que** T n'est pas atteint **Faire**
5: $C \leftarrow$ TournoiBinaire($A_t$)
6: $P_t \leftarrow$ Reproduction($C$)
7: Évaluation par *fine-grained*($P_t$,$A_t$)
8: Sauvegarde de tout les individus avec $R(i)$ dans $A_{t+1}$
9: **Si** $|A_{t+1}| > \overline{N}$ **Alors**
10: Compléter $A_{t+1}$ avec les meilleurs individus de $P_t$ et $A_t$
11: **Sinon**
12: Réduire $A_{t+1}$ par l'opérateur de troncature
13: **Fin Si**
14: $t \leftarrow t + 1$
15: **Fin Tant que**
16: **Retourne** $A$ : Ensemble des solutions non-dominées

Le *MultiObjective Struggle Genetic Algorithm* (MOMGA) (Van Veldhuizen and Lamont, 2000) est composé à chaque itération de trois phases : la phase d'initialisation, la phase primordiale et la phase de juxtaposition. Dans la phase d'initialisation, l'algorithme construit des groupes d'individus à l'aide d'un processus déterministe, par exemple l'initialisation partielle énumérative. La phase primordiale effectue la sélection des individus de la population et réduit la taille de la population si nécessaire. La phase de juxtaposition construit la nouvelle population en utilisant l'opérateur *cut and slice*.

Le *Pareto Envelope-based Selection Algorithm* (PESA) (Corne et al., 2000) utilise une petite population et une archive de grande taille. Pour évaluer la diversité, l'espace du phenotype est divisé, permettant l'évaluation de la répartition de la population de l'archive.

Le *Micro-Genetic Algorithm* (MicroGA) (Coello Coello and Pulido, 2005) a une petite population et un processus de réinitialisation. A chaque itération, une population aléatoire est générée. Cette population est répartie dans la mémoire, qui est divisée en deux parties : remplaçable et non-remplaçable. La partie non-remplaçable ne change pas durant l'exécution et maintient la diversité. La partie remplaçable change à chaque cycle du MicroGA.

Le *MultiObjective Struggle Genetic Algorithm* (MOSGA) (Andersson and Wallace, 2002) a un processus de sélection particulier pour constituer la nouvelle population. Deux individus dans la

population sont sélectionnés pour faire un croisement et une mutation classique pour créer deux enfants. L'enfant est comparé à l'individu dont il est le plus proche dans la population. Si l'enfant domine, alors il remplace l'individu dans la population.

### 1.4.2 Évolution différentielle

L'évolution différentielle, en anglais *Différential Evolution* (DE), (Storn and Price, 1997) est initialement conçue pour l'optimisation dans le domaine continu. Pour améliorer la population au fil des itérations, les différences entre les individus sont utilisées pour créer de nouveaux individus. Bien que la DE utilise des opérateurs communs avec les algorithmes évolutionnaires, la mutation diffère. La mutation tient compte de la répartition de la population dans l'espace pour déterminer les gènes et calculer les nouvelles valeurs.

Pour la première adaptation de la DE aux problèmes multiobjectifs, en anglais *MultiObjective Différential Evolution* (MODE), (Chang et al., 1999) une archive est utilisée pour stocker les solutions non-dominées. Une évaluation qui partage les individus en fonction de leurs caractéristiques est utilisée pour maintenir la diversité de la population. Comme dans la DE, les gènes choisis pour muter et leur nouvelle évaluation dépendent donc de la répartition de la population dans l'espace de solutions.

Le *Pareto-frontier Differential Evolution* (PDE) (Abbass et al., 2001) est une autre variante du DE pour l'optimisation multiobjectif. La population initiale est générée selon une loi Gaussienne. Seules les solutions non-dominées sont retenues pour la reproduction. Trois parents sont sélectionnés aléatoirement pour créer un enfant. L'enfant n'est mis dans la population que s'il domine le premier parent. Le processus est répété jusqu'à ce que la nouvelle population soit complète.

### 1.4.3 Recherche harmonique

La recherche harmonique, en anglais l'*Harmony Search* (HS), (Geem et al., 2001) est inspirée des principes de l'improvisation musicale. Quand les musiciens recherchent l'harmonie en testant plusieurs combinaisons de notes, qu'ils retiennent en mémoire. Contrairement aux EAs, la HS pour la création d'un enfant considère toute la population courante et n'a pas besoin aussi de valeurs initiales pour les variables de décision.

La HS pour les problèmes multiobjectifs, an anglais *MultiObjective Harmony Search* (MOHS), (Ricart et al., 2011) diffère sur le tri des individus en fonction de leurs évaluations aux objectifs. En mono-objectif, le rang est déterminé par la fonction objectif. Par contre en multiobjectif, plusieurs méthodes existent pour attribuer un rang de Pareto, par exemple, en calculant pour chaque individu le nombre d'individus qui le domine. Les non-dominées ont le rang 0 et les solutions dominées ont un rang supérieur ou égal à 1. Et les fronts de Pareto de la population peuvent être déterminés en fonction du rang.

### 1.4.4 Méthode de l'entropie croisée

La méthode de l'entropie croisée, en anglais *Cross Entropy* (CE) (Rubinstein, 1999), est une méthode d'optimisation utilisée pour résoudre les problèmes combinatoires NP-difficiles. La CE est initialement utilisée pour évaluer la probabilité d'événements rares dans les réseaux stochastiques (Rubinstein, 1997). Chaque itération de l'algorithme est composée de deux phases. La première phase, consiste à générer aléatoirement des solutions par une heuristique dont les paramètres sont modifiés par la suite. Les paramètres de l'heuristique sont modifiés, dans la deuxième phase, à partir de l'évaluation des solutions obtenues. Cette phase assure la convergence de l'algorithme.

Une adaptation pour les problèmes multiobjectifs de la CE, en anglais *MultiObjective Cross Entropy* (MOCE) (Perelman and Ostfeld, 2005) pour optimiser un système de distribution d'eau avec les objectifs de minimiser le coût, maximiser la fiabilité, minimiser les risques, minimiser les variations de plusieurs paramètres liés à l'eau, tel que la quantité, la pression, la qualité, etc...

Une étude théorique (Unveren and Acan, 2007), propose le *Cross Entropy for MultiObjective Optimization* (CEMO). Les étapes de la CE sont reprises, mais adaptées pour le multiobjectif. Des solutions sont générées aléatoirement selon une loi de probabilité. Les solutions sont ensuite évaluées et les meilleures solutions sont sélectionnées pour mettre à jour la loi de probabilité. La CEMO a la particularité de diviser le front de Pareto en zone pour permettre la diversification de la recherche.

La CE a aussi une adaptation pour l'optimisation multiobjectif continue (Bekker and Aldrich, 2011) : le *MultiObjective Otpimization Cross Entropy Method* (MOO CEM). Elle est testée sur les benchmarks MOP et ZDT avec les métriques *Generation Distance*, *Spacing*, *Max. Pareto front error*, *Convergence*, le temps d'exécution et le nombre d'individus dominants. Les résultats montrent que le front de Pareto obtenu est proche du front de Pareto optimal.

### 1.4.5 Système immunitaire artificiel

Le système immunitaire artificiel, en anglais *Artificial Immune System* (AIS), (Farmer et al., 1986) (Kursawe, 1991), est une méthode d'optimisation dérivé des algorithmes évolutionnaires. L'AIS s'inspire de la production d'anticorps par un organisme pour répondre à l'intrusion d'un agent pathogène. Les principaux éléments d'un AIS sont : l'antigène, l'antibiotique et l'affinité (voir algorithme 1.4). Les antigènes sont des substances étrangères, la fonction objectif peut-être assimilée aux antigènes. Les antibiotiques sont produits par les cellules-B pour combattre les antigènes ; dans l'algorithme, ils sont des solutions réalisables pour le problème d'optimisation. L'affinité détermine l'efficacité d'antibiotique contre un antigène. Un antibiotique efficace pour lutter contre un antigène a une grande affinité. Pour une solution, l'affinité est considérée comme le résultat de l'évaluation à la fonction objectif. Les principaux opérateurs d'un AIS sont : le clonage, l'hyper-mutation et le remplacement. Les antibiotiques ou solutions en fonction de leur affinité sont sélectionnés aléatoirement pour se reproduire par clonage. Plus une solution a d'affinité plus elle a de chance d'être sélection-

née. L'hyper-mutation est effectuée sur les clones des antibiotiques et étend l'espace de recherche en fournissant à l'algorithme de nouvelles solutions. Le remplacement a sélectionné les meilleurs individus parmi la population courante et ceux issus de l'hyper-mutation. Le nombre d'individus retenus correspond à la taille de la population.

**Algorithme 1.4** Algorithme de l'AIS

**Paramètre** $n$ : Taille de la population d'antibiotique

**Paramètre** $T$ : Critère d'arrêt

1: $PS_0 \leftarrow$ Initialisation($n$)
2: $t \leftarrow 0$
3: **Tant que** T n'est pas atteint **Faire**
4: $C \leftarrow$ SelectionIndividu($n$,$PS_t$)
5: ClonageHypermutation($C$)
6: ÉvaluationAffinité($C$)
7: $PS_{t+1} \leftarrow$ SélectionMeilleurs($n$,$PS_t$,$C$)
8: $t \leftarrow t + 1$
9: **Fin Tant que**
10: **Retourne** Le meilleur antibiotique de $PS_t$.

Les AIS pour les problèmes multiobjectifs, en anglais *MultiObjective Artificial Immune System*, (MOAIS) reprennent les mêmes principes que l'AIS. Le premier *MultiObjective Artificial Immune System* (MOAIS) (Yoo and Hajela, 1999) résout trois problèmes de structure : *Simply-supported I-beam structure* et *l-bar truss structure*. Le *Two-bar truss structure* a pour objectifs de minimiser le poids de la structure et le déplacement vertical du nœud central. Le *Simply-supported I-beam structure* a pour objectifs de minimiser le poids de la structure et la déflexion statique à la mi-envergure. Le *l-bar truss structure* a pour objectifs de minimiser le poids de la structure et le déplacement vertical. Cet algorithme utilise pour l'évaluation des objectifs une fonction linéaire d'agrégation qui permet de représenter l'évaluation et les contraintes en une seule valeur scalaire. Le MOAIS fonctionne comme suit :

1. Initialisation d'une population de structures.
2. Formuler différents vecteurs de poids qui couvrent le front de Pareto.
3. Évaluer chaque structure de la population selon tous les vecteurs de poids.
4. Combiner les notes aux objectifs et contraintes pour obtenir une mesure pour chaque structure et pour chaque vecteur de poids.
5. Pour chaque vecteur de poids, identifier la structure avec la meilleure mesure, ce sont les antigènes.
6. La totalité de la population est définie comme antibiotiques.
7. La procédure suivante est effectuée pour chaque antigène :

— Choisir aléatoirement un antigène.
— Sélectionner aléatoirement quelques antibiotiques (le nombre d'antigènes).
— Chaque antibiotique sélectionné est comparé à l'antigène pour calculer une note basée sur la distance de Hamming.
— L'antigène ayant la meilleure note est gardée.
— La note de cet antigène est ajoutée à son évaluation.

Des MOAIS avec des micro-populations existent tel que le *Micro-population Immune Multiobjective Optimization* (MIMO) (Lin and Chen, 2011). Le MIMO utilise une petite population d'individus pour explorer l'espace de solutions. Un nouvel opérateur de mutation adaptative utilise les évaluations des fonctions objectifs pour agrandir l'espace de recherche et réduire la probabilité d'obtenir des doublons. Une sélection par *fine-grained* est utilisée pour préserver une bonne diversité de la population pour l'itération suivante. Le MIMO est testé sur des problèmes classiques d'optimisation continue (ZDT, DTLZ et WFG) et obtient des résultats meilleurs ou équivalent s à ceux des algorithmes auxquels il est comparé.

### 1.4.6 Algorithme de colonies de fourmis

L'algorithme de colonie de fourmis, en anglais *Ant System* (AS), (Colorni et al., 1992) (Dorigo et al., 2006) est une métaheuristique simulant les colonies de fourmis. Le principe est simple, les fourmis déposent lors de leurs déplacements une substance chimique appelée phéromone. Cette substance influence le comportement des fourmis : elles choisissent les chemins avec le plus de phéromone. En informatique, le AS est un système multi-agents avec peu d'interaction entre les agents. Les fourmis communiquent de façon indirecte par l'intermédiaire des chemins de phéromones.

L'algorithme AS est composé de quatre étapes (voir algorithme 1.5). En premier, les mémoires contenant le taux de phéromones sont initialisées, les valeurs choisies sont arbitraires. Dans la seconde étape, chaque fourmi construit sa solution aléatoirement en fonction des dépôts de phéromones gardés en mémoire. Les positions avec le plus de phéromones ont plus de chance d'être choisies par les agents. En troisième étape, les solutions construites par les agents sont évaluées et la ou les meilleure(s) sont récompensée(s). La phéromone est déposée sur le ou les chemin(s) de(s) meilleure(s) solution(s) et elle s'évapore légèrement sur les autres chemins. La quatrième étape détermine si le critère d'arrêt est atteint. Si le critère d'arrêt est atteint, l'algorithme s'arrête, sinon il continue en retournant à la deuxième étape.

**Algorithme 1.5** Algorithme de l'Ant System (AS)

**Paramètre** $n$ : Taille de la population d'antibiotiques

**Paramètre** $T$ : Critère d'arrêt

1: $M \leftarrow$ InitialisationMémoire
2: $B \leftarrow \oslash$
3: **Tant que** T n'est pas atteint **Faire**
4: $P \leftarrow$ ConstruireSolutions($n$,$M$)
5: Évaluation($P$)
6: DépotEvaporationPhèromones($P$,$M$)
7: $B \leftarrow$ SélectionMeilleurs($B$,$P$)
8: **Fin Tant que**
9: **Retourne** $B$

Le premier AS pour les problèmes multiobjectifs, en anglais *MultiObjective Ant-Q* (MOAQ), (Mariano et al., 1999) utilise une colonie de fourmis pour chaque objectif. Les objectifs sont améliorés les uns après les autres. Une itération se fait en plusieurs phases (voir algorithme 1.6). En premier, les fourmis de chaque colonie cherchent à optimiser un objectif. Chaque colonie est indépendante des autres pour la construction des solutions en utilisant sa mémoire propre. Pendant cette phase la mémoire de la colonie suivante est mise à jour avec les solutions non-dominées trouvées par les fourmis de la famille courante. Dans la deuxième phase, les colonies échangent les informations dans leurs mémoires. Des solutions sont construites à partir des informations dans les mémoires de toutes les colonies. Après évaluation, les meilleures solutions obtenues sont utilisées pour mettre à jour les mémoires par un dépôt de phéromones pour les meilleures solutions et une évaporation dans les mémoires pour l'itération suivante. En dernier, si le critère d'arrêt est atteint, les non-dominées sont retournées par l'algorithme.

**Algorithme 1.6** Algorithme MultiObjective Ant-Q (MOAQ)

**Paramètre** $T$ : Critère d'arrêt

**Paramètre** $m$ : Le nombre de fourmis par population

**Paramètre** $k$ : Le nombre d'objectifs

1: $M \leftarrow$ InitialisationMémoire
2: $B \leftarrow \oslash$
3: **Tant que** T n'est pas atteint **Faire**
4: **Pour** $j = 1 \rightarrow k$ **Faire**
5: $P_j \leftarrow$ ConstruireSolutions($m$,$M_j$)
6: Évaluation($P_j$)
7: $M_{j+1} \leftarrow$ DépotEvaporationPhèromones($P_j$,$M_j$)
8: $B_j \leftarrow$ SélectionMeilleurs($P_j$)
9: **Fin Pour**
10: $ND \leftarrow$ TrouverNonDominés($B$)
11: **Pour** $j = 1 \rightarrow k$ **Faire**
12: DépôtÉvaporationPhèromones($ND$,$M_j$)
13: **Fin Pour**
14: **Fin Tant que**
15: **Retourne** $ND$

### 1.4.7 Optimisation par essaims particulaires

L'optimisation par essaims particulaires, en anglais *Particle Swarm Optimization* (PSO), (Kennedy and Eberhart, 1995) est une métaheuristique qui s'inspire de la chorégraphie du vol des oiseaux en groupes. Le PSO est initialement élaboré pour l'optimisation continue, bien que beaucoup d'adaptation pour l'optimisation combinatoire existent. L'idée est de simuler le mouvement d'un groupe d'oiseaux qui recherche de la nourriture. L'algorithme utilise une population et une fonction d'évaluation similaire à celle utilisée pour les algorithmes évolutionnaires. Durant l'exécution de l'algorithme, chaque individu (ou particule) a son déplacement est affecté par la meilleure solution qu'il a visité et la meilleure solution explorée par la population (voir algorithme 1.7). Les valeurs générées aléatoirement pour calculer la nouvelle position de chaque particule est affectée par les poids donnés aux meilleurs optimums locaux et globaux. Le PSO permet aux individus de bénéficier de leurs expériences passées, tandis que dans les EAs les individus utilisent uniquement les informations contenues dans la population courante. Il utilise aussi la notion de vol dans l'hyperespace des solutions.

**Algorithme 1.7** Algorithme *Particle Swarm Optimization* (PSO)

**Paramètre** $T$ : Critère d'arrêt

**Paramètre** $n$ : Taille de la population

**Paramètre** $C_1$ : Poids de la meilleure solution globale

**Paramètre** $C_2$ : Poids de la meilleure solution locale

**Paramètre** $w$ : Inertie

1: $P \leftarrow$ InitialisationPositionVitesse($n$)
2: Évaluation($P$)
3: $BG \leftarrow$ SélectionMeilleurGlobal($P$)
4: $BL \leftarrow$ SélectionMeilleurLocal($P$)
5: **repeat**
6: **Pour** $i = 1 \rightarrow n$ **Faire**
7: $V_i = w \times V_i + C_1 \times rand_1 \times (BL_i - P_i) + C_2 \times rand_2 \times (BG - P_i)$
8: $P_i = P_i + V_i$
9: Évaluation($P_i$)
10: $BL_i \leftarrow$ SélectionMeilleurLocal($P_i$)
11: **Fin Pour**$BG \leftarrow$ SélectionMeilleurGlobal($P$)
12: **until** T n'est pas atteint
13: **Retourne** $BG$

Plusieurs adaptations du PSO aux problèmes multiobjectifs, en anglais *MultiObjective Particle Swarm Optimization* (MOPSO), ont été proposées (Parsopoulos and Vrahatis, 2002), (Ray and Liew, 2002) et (Coello Coello and Lechuga, 2002) ; elles utilisent toutes les principes utilisés par le PSO (voir algorithme 1.8). Pour chaque individu, les meilleurs optimums, global et local, sont sélectionnés aléatoirement si plusieurs candidats existent. Donc, l'une des principales difficultés pour le MOPSO est de choisir la meilleure solution dominante par rapport à une autre. Plusieurs techniques existent telles que les méthodes de l'hypercube et du sigma (Yang et al., 2009) en remplacement d'une sélection aléatoire, permettant une nette amélioration des résultats.

**Algorithme 1.8** Algorithme *MultiObjectiveParticle Swarm Optimization* (MOPSO)

**Paramètre** $T$ : Critère d'arrêt
**Paramètre** $n$ : Taille de la population
**Paramètre** $C_1$ : Poids de la meilleure solution globale
**Paramètre** $C_2$ : Poids de la meilleure solution locale
**Paramètre** $w$ : Inertie

1: $P \leftarrow$ InitialisationPositionVitesse($n$)
2: Évaluation($P$)
3: $BG \leftarrow$ SélectionMeilleurGlobal($P$)
4: $BL \leftarrow$ SélectionMeilleurLocal($P$)
5: **repeat**
6: **Pour** $i = 1 \rightarrow n$ **Faire**
7: $V_i = w \times V_i + C_1 \times rand_1 \times (BL_i - P_i) + C_2 \times rand_2 \times (BG_i - P_i)$
8: $P_i = P_i + V_i$
9: Évaluation($P_i$)
10: $BL_i \leftarrow$ SélectionMeilleurLocal($P_i$)
11: **Fin Pour**
12: $ND \leftarrow$ SélectionNonDominée($P_i$)
13: **Pour** $i = 1 \rightarrow n$ **Faire**
14: $BG_i \leftarrow$ SélectionMeilleurGlobal($ND$)
15: **Fin Pour**
16: **until** T n'est pas atteint
17: **Retourne** $ND$

### 1.4.8 Algorithme à colonies d'abeilles

Le PSO étant initialement prévu pour l'optimisation continue, une autre famille d'algorithme utilise la notion de vol dans l'espace de solutions, pour les problèmes combinatoires. L'algorithme à colonie d'abeilles, en anglais *Artificial Bees Colony* (ABC), (Pham and Ghanbarzadeh, 2007) et (Pham et al., 2006) s'inspire du vol des abeilles à la recherche de fleurs pour butiner. L'ABC explore le voisinage des sources de nourriture pour converger, et, pour diversifier, des solutions sont créées aléatoirement (voir algorithme 1.9). L'algorithme commence par la création aléatoire des sources de nourriture et leur évaluation. Dans une première phase d'amélioration, une recherche locale est appliquée sur des sources de nourriture sélectionnées aléatoirement. En second, une recherche locale est faite sur des sources de nourriture sélectionnées au hasard, mais les meilleures évaluations ont plus de chance d'être choisies. Après, le nombre de sources de nourriture est réduit en ne gardant que les meilleures solutions. La troisième étape diversifie l'exploration en créant des solutions aléatoirement, jusqu'à atteindre la taille de la population initiale. Après, si le critère d'arrêt n'est pas atteint, l'algorithme revient à la première phase. Sinon, la meilleure solution est retournée par l'algorithme.

**Algorithme 1.9** Algorithme *Artificial Bees Colony* (ABC)

**Paramètre** $m$ : Nombre de sources de nourriture sélectionnées pour la recherche de voisinage

**Paramètre** $e$ : Nombre d'abeilles pour visiter les meilleurs sources de nourriture

**Paramètre** $n$ : Nombre d'abeilles pour trouver de nouvelles sources de nourriture

**Paramètre** $T$ : Un critère d'arrêt

1: Initialisation de la population avec des sources de nourriture aléatoires
2: évaluation de la population
3: **Tant que** $T$ n'est pas atteint **Faire**
4: Sélectionner $m$ sources de nourriture pour faire des recherches locales
5: Visiter le voisinage des $e$ meilleurs sources de nourriture
6: Sélectionner les meilleurs voisinages visités par les abeilles
7: Créer $n$ sources de nourriture aléatoirement et les évaluer pour compléter la nouvelle population
8: **Fin Tant que**
9: **Retourne** Meilleure source de nourriture

Plusieurs versions de l'ABC sont proposées pour l'optimisation multiobjectif, le *MultiObjective Artificial Bee Colony* (MOABC), (Pham and Ghanbarzadeh, 2007) et (Akbari et al., 2012). Le MOABC reprend les principes de l'ABC (voir algorithme 1.10), il est composé de trois phases, deux phases pour converger *Employed Bees* et *Onlooker Bees* et une phase pour diversifier *Scout Bees*.

La phase *Employed Bees* consiste à améliorer des individus de la population par une recherche locale. Dans la phase *Onlooker Bees*, des sources de nourriture sont améliorées aussi par une recherche locale, mais les individus avec une bonne évaluation ont plus de chance d'être sélectionnés. La recherche locale utilisée dans les phases *Employed Bees* et *Onlooker Bees* est définie par :

$$x_{new}^{j} = w.rand[-1,1].(x_i^j - x_k^j) \quad \forall j \in \{1,...,n\}$$

où $w$ est l'inertie donnée au déplacement, $x_i = \{x_{i1}, x_{i2}, ..., x_{in}\}$ est le vecteur représentant la $i^{eme}$ source de nourriture et $n$ est le nombre de dimensions du problème. La différence entre les phases *Employed Bees* et *Onlooker Bees* est dans le processus pour choisir la solution $x_k$. Les *employed bees* choisissent aléatoirement la solution $x_k$ en suivant une loi de probabilité uniforme. La loi de probabilité utilisée par les *onlooker bees* est la suivante :

$$p_k = \frac{fit(\vec{x}_k)}{fit(\sum_{m=1}^{NS} fit\,(\vec{x}_m))}$$

où $NS$ est le nombre de sources de nourriture, $fit(\vec{x}_m)$ est l'évaluation de la solution $x_m$ et $NS$ est le nombre de solutions. Lors la phase *Scout Bees*, un petit nombre de sources de nourriture sont remplacées par d'autres créées aléatoirement. La mise à jour d'une archive contenant les solutions non-dominées (*Update Archive*) est faite après la phase *Scout Bees*.

**Algorithme 1.10** Algorithme *MultiObjective Artificial Bee Colony* (MOABC)

**Paramètre** $T$ : Un critère d'arrêt

**Paramètre** $n$ : Taille de la population

**Paramètre** $e$ : nombre de *employed bees*

**Paramètre** $o$ : nombre de *onlooker bees*

**Paramètre** $s$ : nombre de *scoot bees*

1: $P \leftarrow$ Initialisation($n$)
2: **repeat**
3: Phase Employed_Bees($P$,$e$)
4: Phase Onlooker_Bees($P$,$o$)
5: Phase Scout_Bees($P$,$s$)
6: $A \leftarrow$ Update Archive($P$)
7: **until** T atteint
8: **Retourne** $A$

### 1.4.9 Scatter search

La recherche dispersée, en anglais *Scatter Search* (SS), (Glover, 1977) (Glover et al., 2000) est une méthode pour la programmation entière. Le SS est une méthode de recherche qui utilise une série de différentes initialisations pour produire de nouvelles solutions. Pour cela, un ensemble de solutions est utilisé comme référence. Le SS utilise une approche déterministe au lieu d'utiliser le hasard. L'idée est d'identifier les combinaisons de points convexes dans l'ensemble de référence et de les combiner avec les sous-ensembles de références initiaux pour obtenir de nouvelles sous-régions à explorer. A l'étape suivante, les points à l'intérieur des sous-régions sont examinés pour rechercher de nouvelles zones à explorer.

La première adaptation du SS pour l'optimisation multiobjectif, le *Multiple Criteria Scatter Search* (MCSS), (Beausoleil Delgado, 2001) utilise la recherche tabou pour générer l'ensemble initial, appelé le *MultiStart Tabou Search* (MSTS) et les individus sont classés par front de Pareto. Une autre version, le *MultiObjective Scatter Search* (MOSS) Beausoleil (2006) est une amélioration du MOSS pour les problèmes non-linéaires. Le MOSS crée aléatoirement un ensemble de solutions initiales avec le MSTS. À chaque itération, les solutions non-dominées sont identifiées pour servir de points de référence. De nouvelles solutions sont créés en combinant les solutions de référence et ensuite elles sont évaluées. Les nouvelles solutions non-dominées sont ensuite identifiées dans l'ensemble de référence et les solutions nouvellement créées. Pour diversifier ce nouvel ensemble, le MSTS est utilisé pour générer de nouvelles solutions à partir des solutions non-dominées. Les meilleures solutions sont ensuite retenues dans la population de la génération suivante.

L'un des plus efficaces des SS pour l'optimisation multiobjectif est le *Archive-Based hYbrid Scatter Search* (AbYSS) (Glover, 1998). L'AbYSS peut aussi résoudre des problèmes d'optimisation

continue. Les particularités de l'AbYSS se portent de cinq méthodes : la diversification de génération, l'amélioration de la population, la mise à jour de l'ensemble de référence, la génération de sous-ensembles et la combinaison de solution.

La méthode de diversification de génération a pour objectif de créer un ensemble de solutions bien réparties dans l'espace de recherche. C'est une méthode simple basée sur la division de la distance entre chaque variables en un nombre déterminé de sous-intervalles de taille égale. La valeur de chaque variables de décision est calculée en deux étapes. En premier, un sous-intervalle d'une variable est choisi aléatoirement. La probabilité de sélectionner l'intervalle est inversement proportionnelle au nombre de fois où il a été déjà choisi. En second, une valeur est générée aléatoirement dans un intervalle précédemment sélectionné. Ceci est répété pour toutes les variables de décision de la solution à créer.

La méthode d'amélioration utilise la recherche locale pour améliorer les nouvelles solutions obtenues par la méthode de diversification de génération et la combinaison de solutions. Pour chaque solution, une exploration de voisinage est faite pour trouver une solution dominant l'initiale.

La méthode de mise à jour de l'ensemble de référence garantit la présence équilibrée de solutions de qualité et diversifiées. L'ensemble de référence est divisé en deux sous-ensembles de tailles prédéterminées pas forcément égales. Le premier contient les solutions de meilleure qualité (non-dominée). Le second est composé de solutions garantissant la diversité de l'ensemble de référence. Les solutions du second ensemble sont choisies parmi ceux qui ont la plus grande distance euclidienne avec ceux du premier.

La méthode de génération de sous-ensembles génère chaque nouvelles solutions à partir de la combinaison d'un individu de l'ensemble des solutions non-dominées et d'un autre appartenant à celui des solutions diversifiées. Combiner une solution dans chaque sous-ensemble équilibre l'intensification et la diversification de l'algorithme.

La méthode de construction de solutions cherche des combinaisons linéaires avec les solutions de l'ensemble de référence. L'opérateur de combinaison utilisé est le *Simulated Binary Crossover* (SBX) (Agrawal and Deb, 1994). Lors du croisement, l'avantage du chromosome codé en binaire est sa capacité à transmettre de larges blocs d'information à ses enfants. Le SBX est une adaptation du croisement binaire pour les chromosomes codé en nombres réels.

Les résultats montrent que l'AbYSS obtient de meilleurs résultats que les très populaires NSGA-II et SPEA2 sur les benckmarks ZDT, WFG et DTLZ.

### 1.4.10 Apprentissage par renforcement distribué

Le *Q-Learning* (QL) (Watkins and Dayan, 1992) utilise une technique d'apprentissage pour que les individus de sa population se déplacent efficacement. Le QL est similaire à une méthode de programmation dynamique. Cet algorithme fonctionne en améliorant successivement ses connaissances

à chaque itération. Les solutions explorées sont évaluées et ce qui permet ensuite de mettre à jour la mémoire. La meilleure solution explorée par l'algorithme est en sortie à la fin de l'exécution.

L'adaptation du QL pour les problèmes multiobjectifs, le *Multiobjective Distributed Q-Learning* (MDQL) (Mariano and Morales, 2000a) (Mariano and Morales, 2000b) (Mariano and Morales, 2000c) suit la même trame algorithmique en utilisant une population pour chaque individu. Les solutions obtenues par les individus d'une population sont comparées avec les solutions obtenues par les autres populations. Le principe est d'améliorer le front de Pareto en respectant tous les objectifs. Les solutions non-dominées sont utilisées pour mettre à jour les connaissances et sauvegardées pour une future utilisation. Lorsque le critère d'arrêt, la liste des solutions non-dominées est retournée à l'utilisateur.

## 1.5 Méthodes approchées sans population

La plupart des algorithmes sans population utilisent un seul individu pour explorer l'espace de recherche. D'autres n'en utilise pas comme la méthode de Nelder-Mead. La plupart sont des améliorations de la Recherche Locale qui est une méthode. Mais cette méthode est tombe facilement dans un minimum local. La recherche taboue et le recuit simulé autorisent certains mouvements dégradant la solution courante pour sortir des minimums locaux. Le *Greedy Randomized Adaptive Search Procedure* repart d'une solution différente à chaque itération. Les réseaux de neurones améliorent à chaque itération la solution courante. La méthode de Nelder-Mead modifie sa zone de recherche jusqu'à la réduire à un point.

### 1.5.1 Recherche tabou

La recherche tabou, en anglais *Tabu Search* (TS) (Glover, 1986), est une métaheuristique qui utilise une seul individu pour explorer l'espace de solutions. Cette méthode reprend des éléments de travaux précédents (Glover et al., 1993) et (Hansen, 1986). L'optimum de la recherche tabou est approché itérativement. À chaque itération, un mouvement est effectué en partant de la solution courante vers une solution voisine autorisée avec une meilleure évaluation. La TS fonctionne comme une recherche locale. Cependant pour sortir des minimums locaux, la TS autorise les mouvements vers des solutions qui sont moins favorables au cas où il n'y pas d'amélioration possible. Pour cela, la TS interdit une liste de mouvements pour éviter un cycle. Les mouvements interdits sont stockés dans une structure appelée liste tabou. La liste tabou est mise à jour à chaque itération en fonction de règles déterminées par l'utilisateur. Grâce a cette mémoire la TS sort des minimums locaux.

Plusieurs versions pour l'optimisation multiobjectif de la TS existent, (Hertz et al., 1994), (Hansen, 1997) et (Hansen, 2000).

Dans (Hertz et al., 1994) trois approches sont traitées pour les problèmes mutliobjectifs. La

première fait une somme pondérée des objectifs, et fonctionnent comme un TS mono-objectif. La seconde technique traite lexicographiquement les objectifs. Dans la troisième, la méthode $\epsilon$-contrainte est utilisée pour traiter les objectifs séquentiellement. Lorsqu'un objectif est en cours d'optimisation les valeurs concernant les autres objectifs leurs valeurs fixées par des constantes.

Le *MutliObjective Tabu Search* (MOTS) (Hansen, 1997) génère aléatoirement des solutions de départ pour l'algorithme. À chaque solution, un vecteur de pondération des objectifs est affecté en utilisant la métrique Tchebycheff $\lambda - weighted$. Le principe est d'attribuer ces poids de telle sorte que les solutions non-dominées soient réparties uniformément sur le front de Pareto. Les nouvelles solutions sont générées par la variation des poids donnés à chaque objectif. L'exploration similaire au TS, le meilleur voisin est choisi, en vérifiant qu'il n'est pas dans la liste tabou. Les solutions non-dominées sont archivées tout au long du processus.

Le *TAbu search for MultiObjective Combinatorial Optimization* (TAMOCO) (Hansen, 2000) est une version améliorée du MOTS adaptée aux problèmes combinatoires. Le TAMOCO reprend les mêmes principes du MOTS en utilisant la manipulation d'un vecteur de coefficients pour les objectifs pour trouver le front de Pareto.

### 1.5.2 Recuit simulé

La méthode du recuit pour forger les métaux a inspiré une métaheuristique d'optimisation le *Simulated Annealing* (SA) (Reeves, 1993). Le recuit consiste à appliquer sur une pièce métallique un cycle de chauffage pour modifier les propriétés physiques du métal. En premier lieu, la température d'un matériau est augmentée au stade où les atomes sont libres de bouger. Dans une deuxième phase, la température est abaissée pour forcer les atomes a ce replacer eux-mêmes dans une autre position (processus de cristallisation). Pendant cette dernière phase, l'énergie du solide est minimisée. Le refroidissement est important pour le processus. Si le solide est refroidi trop rapidement ou si la température initiale est trop basse, le solide ne peut pas cristalliser et n'obtient pas les propriétés voulues. Le premier SA (Reeves, 1993) utilise une seule solution. Il composé de plusieurs cycles qui sont composés de plusieurs itérations. À chaque itération, l'algorithme explore le voisinage de la solution courante. Si la "température" le permet la solution voisine explorée est acceptée pour servir à la prochaine itération. Si une solution est meilleure alors elle est automatiquement sélectionnée, sinon elle est sélectionnée en fonction d'une loi de probabilité. Quand le nombre d'itérations est atteint, la "température" est abaissée pour le prochain cycle. L'algorithme s'arrête lorsque la condition d'arrêt est atteinte, elle peut être le nombre de cycle ou le temps.

La première adaptation du SA pour les problèmes multiobjectifs, en anglais *MultiObjective Simulated Annealing* (MOSA) (Ulungu et al., 1999), reprend les lignes générales utilisées pour l'optimisation mono-objectif. À la différence du SA qui fournit une seule solution optimale déterminée sur un seul critère, le MOSA va construire une liste de solutions non-dominées. Cette liste de solutions non-dominées en sortie du MOSA est l'approximation du front de Pareto obtenue.

L'*Archive Multi Objective Simulated Annealing* (AMOSA) (Bandyopadhyay et al., 2008) est proposé pour résoudre efficacement les problèmes avec de nombreux objectifs (4, 5, 10 et 15 objectifs). C'est un MOSA modifié par l'ajout d'une archive, qui permet d'intégrer un nouveau concept de dominance afin de déterminer le choix d'une nouvelle solution. L'AMOSA reprend les principes du SA, mais ajoute une archive pour stocker les solutions non-dominées déjà visitées. Cette archive a deux limites, la limite basse donne le nombre de solutions non-dominées voulues par l'utilisateur. À chaque fois qu'une solution non-dominée est trouvée, elle est ajouté à l'archive. Si la limite haute est atteinte alors le nombre de solutions dans l'archive est réduit à la limite basse de l'archive. L'AMOSA utilise l' *amount domination* pour choisir une solution dominée pour la prochaine itération, plus la solution est dominée par une solution courante moins elle a de chance d'être sélectionnée. Bandyopadhyay et al. (2008) comparent l'AMOSA aux algorithmes MOSA, PAES et NSGA-II sur les benchmarks SCH, ZDT et DTLZ. Les résultats montrent que l'AMOSA donne de meilleurs résultats en qualité de convergence et diversité sur le front de Pareto.

### 1.5.3 Procédure de recherche d'adaptation aléatoire gloutonne

La méthode *Greedy Randomized Adaptive Search Procedure* (GRASP) (Feo and Resende, 1995) est une métaheuristique à démarrages multiples. Chaque itération est composée de deux phases. La première consiste à construire aléatoirement une solution faisable par un algorithme glouton. Dans la seconde phase, la solution est améliorée par une recherche locale jusqu'à atteindre un minimum local. Ces deux phases sont répétées tant que le critère d'arrêt n'est pas atteint. La meilleure solution explorée est en sortie de l'algorithme.

Le *MultiObjective GRASP* (MOG) (Reynolds et al., 2009) est la version multiobjectif du GRASP. La recherche locale est appliquée pour les fronts de Pareto, à la place des individus. Quand une solution améliore la qualité du front de Pareto, elle est conservée. L'algorithme glouton peut-être appelé, plusieurs fois avant que d'appliquer la recherche locale. Ceci permet à la recherche locale de partir de bonnes solutions. Une liste des solutions non-dominées est retournée par l'algorithme.

### 1.5.4 Méthode de Newton

La méthode de Newton, en anglais *Newton's Method for Multiobjective Optimization* (NMMO) (Fliege et al., 2009) est une métaheuristique issue d'une adaptation d'un algorithme efficace du même nom pour trouver numériquement une approximation précise de la racine d'un polynôme. Contrairement à d'autres méthodes qui utilisent un vecteur de coefficients pour les objectifs avant de les additionner ou un classement lexicographique, la NMMO utilise la méthode du gradiant pour trouver la pente optimale pour la descente vers les optimums. Ceci implique que la fonction objectif du problème doit être deux fois dérivable. La convergence de la NMMO est prouvée et si les dérivées du deuxième degré sont lipschitzienne, la convergence est quadratique.

À chaque itération de l'algorithme, une minimisation est effectuée à partir d'une solution dite non-stationnaire (non-optimum local) construite au hasard. Une direction de recherche est calculée pour trouver une meilleure solution. Si cette solution est un optimum local, la recherche s'arrête, sinon une nouvel direction de recherche est calculée. À la fin de l'algorithme les solutions non-dominées sont fournies.

La méthode *Quasi-Newton* (Qu et al., 2011) résout les problèmes multiobjectifs non-convexes. La méthode ne converge que si les conditions initiales sont remplies.

### 1.5.5 Réseau de neurones artificiels

Les réseaux de neurones artificiels, en anglais *Neural Networks* sont aussi utilisés pour résoudre des problèmes d'optimisation (Tank and Hopfield, 1986). Le réseau proposé est composé de plusieurs circuits basiques (neurones) qui suivent des règles de calculs collectives qui respectent les contraintes et l'objectif du problème. La solution courante est améliorée à chaque itération en étant réintroduite à l'entrée du réseau à chaque itération.

Le premier réseau de neurones pour l'optimisation multiobjectif utilise les modèles de Hopfield étendus (Balicki et al., 1998). Cet algorithme utilise un vecteur de coefficients. Chaque coefficient est affecté à un objectif. La somme des objectifs multiplié par leur coefficient est la note obtenue par la solution. Lorsque la solution optimale est trouvée le vecteur est modifié et l'algorithme est relancé. À chaque fois qu'une nouvelle solution non-dominée est trouvée, l'approximation du front de Pareto est complété. Ce réseau peut résoudre des problèmes linéaires ou quasi-quadratiques.

### 1.5.6 Méthode de Nelder-Mead (Méthode du simplex)

La Méthode de Nelder-Mead ou *Nelder-Mead method* (Nelder and Mead, 1965) exploite le concept du simplexe qui est un polytope de N+1 sommets dans un espace à N dimensions. L'algorithme part d'un simplexe initial au cours des itérations ce dernier subit des transformations simples tel que la déformation et le déplacement. Le simplex va se réduire progressivement jusqu'à ce que ses sommets atteignent une solution commune qui est un minimum local.

Des versions multiobjectifs ont été depuis proposées le *Non-dominated Sorting - Simplex algorithm* (NS-Sipmlex) (Ruan et al., 2010) et *Nonlinear Simplex Search for Multiobjective Optimization* (NSS-MO) (Martinez et al., 2011). Le NS-Simplex utilise un tri des solutions non-dominées pour assurer la génération des fronts de Pareto et accélérer la convergence de l'algorithme. Le NSS-MO adopte un schéma de recherche avec un simplex non-linéaire pour obtenir un front de Pareto diversifié. La recherche est dirigée avec un ensemble de vecteurs de coefficients pour chaque objectif. Cet ensemble doit être suffisamment bien réparti pour assurer un front de Pareto de bonne qualité. Ces vecteurs permettent une scalarisation des objectifs du problème et permet au NSS-MO de fonctionner par la suite comme la méthode de Nelder-Mead. L'algorithme est relancé pour chaque

vecteur de poids, ce qui permet de constituer une approximation du front de Pareto du problème multiobjectif.

## 1.6 Hybridations

Le principe de l'hybridation est simple, améliorer les performances de méthodes citées précédemment. Une hybridation peut être la collaboration entre deux métaheuristiques, par exemple un EA avec une recherche locale, un PSO et ACO. L'ajout d'un module qui améliore les performances de la méthode tel qu'un Contrôleur par Logique Floue ou une archive. D'autres comme la coévolution utilise plusieurs populations d'individus différents. Certaines hybridations comme la décomposition ne sont faites que pour les problèmes multiobjectifs.

### 1.6.1 Recherche locale

La recherche locale, en anglais *Local Search* (LS) est une métaheuristique conç ue pour résoudre les problèmes NP-difficiles (Hoos and Stützle, 2004). Une LS part d'une solution et en explore le voisinage itérativement pour trouver une meilleure solution. Donc, avant d'appliquer une LS, un voisinage doit être défini dans l'espace de solutions. Par exemple, dans le cas d'un problème de tournée de véhicule deux solutions peuvent être voisines si une seule ville n'est pas à la même place dans le parcours du véhicule. Le *Hill Climbing* (HC) est une LS qui lorsqu'elle trouve une meilleure solution en explore le voisinage pour trouver une nouvelle solution améliorante. La condition d'arrêt d'une LS peut être le temps, quand une meilleure solution est trouvée ou lorsqu'elle ne trouve pas de solutions améliorantes depuis un nombre d'itérations. Le désavantage de la recherche locale est qu'elle peut s'arrêter alors que la meilleure solution trouvée n'est pas un optimum global.

Une recherche locale ne donnant pas fréquemment l'optimum global, elle est rarement utilisée seule, mais par contre elle est à l'origine de nombreuses autres méthodes tels que le TS ou le SA. Elle est aussi souvent utilisée pour hybrider d'autres algorithmes tels que les EAs ou les PSOs. Les EAs hybridés avec la recherche sont appelés aussi algorithmes mimétiques (Moscato et al., 1989). La LS peut-être appliquée à toutes les itérations ou périodiquement. L'utilisateur doit choisir sur quels types de solutions s'applique la LS, par exemple les meilleures solutions. En général, l'utilisation d'un recherche locale permet pour le moins l'accélération de la convergence, mais aussi des résultats.

Pour les problèmes multiobjectifs, la recherche locale rencontre une difficulté lorsque deux solutions sont non-dominées entre elles. Plusieurs politiques se pratiquent tel que choisir au hasard ou garder systématiquement l'ancienne ou la nouvelle solution.

### 1.6.2 Contrôleur par Logique Floue

La logique floue s'appuie sur la théorie mathématique des ensembles flous (Zadeh, 1965). Les ensembles classiques d'appartenance ne définissent que grossièrement un état. Les ensembles flous permettent de nuancer le degré d'appartenance. La logique classique attribue à un événement la valeur 0 ou 1, 0 signifie échec et 1 une réussite. Mais des états intermédiaires peuvent exister, la logique floue mesure le degré d'appartenance d'un événement à chacun des ensembles (0 ou 1). Par exemple, si deux personnes mesurent respectivement 1,70 m et 1,90 m et sont considérées comme de taille moyenne et grande. Avec les ensembles classiques, une troisième personne de taille 1,80 m sera considérée soit de taille moyenne, soit de taille grande. Ceci est arbitraire et de l'information est perdue car il existe la même différence de taille avec les deux autres personnes. La logique floue introduit un degré d'appartenance, dans l'exemple elle est 50% de taille moyenne et 50% de taille grande. En recherche opérationnelle, il existe plusieurs collaborations entre les métaheuristiques et la logique floue.

Le *Fuzzy Preference-Based Multiobjective Optimization Method* (FPMOM) (Ramakrishnan and Hasan, 2013) utilise la logique floue pour choisir la solution non-dominée avec le meilleur compromis entre les objectifs. Les notes obtenues aux objectifs par une solution non-dominée sont pris en compte comme paramètres d'entré. La logique floue permet de choisir une autre solution avec un meilleur compromis. Le SPEA est l'algorithme évolutionnaire choisit pour les expérimentations. Les simulations montrent une amélioration de la gestion des objectifs contradictoires.

Le *Chaotic Improved Honey Bee Mating Optimization* (CIHBMO) (Niknam, 2011) utilise la logique floue lors de l'évaluation des objectifs. Dans l'article le CIHBMO est appliqué à la gestion de générateurs distribués, connectés à un réseau électrique. Le problème est de calculer pour le lendemain : les puissances active et réactive de générateurs distribués connectés à un réseau électrique et aussi la position des transformateurs. Les objectifs sont : la minimisation du coût de l'énergie produite et perdue et de la distorsion du signal de tension. La logique floue intervient après le calcul des objectifs d'une solution, la note obtenue par la solution à l'objectif flou est la plus petite valeur des objectifs. Les simulations montrent une amélioration des performances par rapport au HBMO classique.

Le *Hybrid Spiral-Dynamic Bacteria Chemotaxis* (HSDBC) (Nasir and Tokhi, 2015) améliore le signal de commande envoyer à un contrôleur flou qui commande un robot manipulateur. Les résultats montrent une amélioration du temps de réaction, du temps de stabilisation et la diminution du nombre d'erreurs pour les manipulations par rapport à d'autres algorithmes.

L'hybridation avec un contrôleur par logique floue ou *Fuzzy Logic Controler* (FLC) permet d'adapter dynamiquement la convergence et la diversification d'une métaheuristique. L'utilisation d'un FLC permet d'accélérer la convergence et les résultats de la métaheuristique. Le FLC peut être utilisé avec toutes les métaheuristiques, mais il s'adapte très bien avec les algorithmes évolutionnaires (Ah King et al., 2004). Avec un algorithme génétique, les taux de croisement et de mutation sont mise

à jour périodiquement pour obtenir la meilleure convergence possible. À la fin de chaque période, la variation de chaque taux est calculée selon les caractéristiques de la population. L'utilisateur choisit les méthodes d'évaluation à l'entrée du FLC. Le FLC calcule la variation en suivant des tables de règles qui définissent l'influence de chaque paramètre mis à l'entrée du FLC.

Le FLC est aussi utilisé pour les méthodes multiobjectifs (Lau et al., 2009). Les métriques pour évaluer les métaheuristiques pour l'optimisation multiobjectif (Ziztler et al., 2003) sont utilisées à l'entrée du FLC. Les résultats montrent que le FLC améliore les performances des méthodes multiobjectifs testées.

Une autre application du réglage dynamique de paramètres, un *Bacterial Foraging Algorithm* (BFA) (Tabatabaei and Vahidi, 2011) est adapté pour résoudre le problème d'emplacement et de taille de condensateur et de dérivateurs dans un système de distribution électrique. Les objectifs sont de réduire la force des pics de puissance et de minimiser les pertes d'énergie, ainsi qu'améliorer les caractéristiques des nœuds de tension. Dans le BFA, la logique floue est utilisée pour choisir la ligne du réseau à modifier pour améliorer les notes obtenues aux objectifs.

### 1.6.3 Décomposition pour les problèmes multiobjectifs

Le principe de la décomposition est de diviser un problème multiobjectif en plusieurs sous-problèmes en utilisant plusieurs vecteurs de coefficients pour la fonction objectif. Chaque objectif se voit affecter un coefficient et chaque sous-problème devient mono-objectif. Plusieurs méthodes existent pour cette réduction : la sommation des objectifs, la méthode Tchebycheff et la méthode par intersection des limites. Le *Multiobjective Evolutionary Algorithm Based on Decomposition* (MOEA/D) (Zhang and Li, 2007) utilise un algorithme évolutionnaire hybridé par la décomposition. Plusieurs vecteurs de coefficients sont définis pour explorer efficacement l'espace de solutions. L'algorithme commence par l'initialisation d'une population avec un individu pour chaque vecteur de coefficients. La distance euclidienne entre chaque vecteur est calculée et les plus proches sont identifiés. Pour chaque sous-problème, une nouvelle solution est créée, les parents sont issus des sous-problèmes dont les vecteurs sont proches de celui du sous-problème courant. La nouvelle solution est réparée et améliorée. Ensuite, elle est évaluée par la fonction objectif modifiée, si elle est meilleure, alors elle remplace l'ancienne solution du sous-problème. L'archive contenant les solutions non-dominées est mise à jour. L'algorithme retourne à la création de solutions tant qu'il n'a pas atteint son critère d'arrêt. Cet algorithme fonctionne aussi bien pour des problèmes continus que combinatoires. Il permet une nette amélioration des résultats, mais reste difficile à paramétrer.

### 1.6.4 Co-évolution

Dans la nature des organismes ont des relations symbiotiques ou de compétition avec d'autres organismes. L'endosymbiose est la "symbiose dans laquelle un symbiote habite dans le corps de son

partenaire symbiotique". La coévolution est le changement de la composition génétique d'une espèce en réponse à la modification génétique d'une autre. En résumé, la coévolution est le changement réciproque d'espèces qui interagissent entre elles. L'interaction peut-être positive ou négative, c'est-à-dire qu'il peut y avoir collaboration ou compétition entre les espèces concernées.

Les algorithmes coévolutionnaires ont utilisés plusieurs principes, énoncés précédemment, de la coévolution impliquant plusieurs populations différentes. La présence de plusieurs populations (espèces) différentes et l'évaluation d'un individu qui dépend des individus présents dans les autres espèces sont les caractéristiques principales des algorithmes coévolutionnaires. La caractéristique d'un algorithme coévolutionnaire est sa composition de plusieurs populations (espèces) différentes et l'évaluation d'un individu dépend des individus présents dans les autres espèces

Il existe deux classes d'algorithmes coévolutionnaires. Les algorithmes coévolutionnaires avec compétition, dans ce cas les évaluations dépendent des affrontements entre les individus des différentes espèces (Paredis, 1998). Cette approche est souvent utilisée dans les jeux. Dans la deuxième classe les différentes espèces utilise la collaboration. Les individus sont évalués en fonction de la qualité de leur collaboration avec ceux des autres espèces (Potter and De Jong, 1994). L'opération est souvent utilisée dans l'optimisation. Le *Cooperative Coevolutionnary Genetic Algorithms* (CCGA) est l'un des premiers algorithmes co-évolutionnaires utilisant une approche symbiotique. Le CCGA reprend les principes des algorithmes coévolutionnaires coopératifs. La valeur des individus composant une espèce dépend de sa capacité à échanger des informations avec ceux des autres espèces. Pour former une solution, il faut sélectionner au moins un individu de chaque espèce. Cette solution est ensuite évaluée. Les individus sont notés en fonction de la qualité de la solution qu'ils composent.

Les algorithmes coévolutionnaires ont été adaptée pour les problèmes multiobjectifs par exemple le *Genetic Symbiosis Algorithm* (GSA) (Jiangming et al., 2000). Le GSA pour les problèmes multiobjectifs a deux caractéristiques.

Des paramètres sont introduits pour évaluer la symbiose entre les individus. En premier lieu, l'algorithme modifie l'évaluation d'un individu en fonction de sa capacité à collaborer avec ceux des autres espèces. Le deuxième paramètre évalue la collaboration globale d'un individu avec ceux des autres espèces en sommant les notes obtenues pour chaque collaboration.

Certains sont des adaptations d'algorithmes évolutionnaires multiobjectifs tels que le NSCCGA qui est une combinaison du CCGA (Potter and De Jong, 1994) et du NSGA-II (Deb et al., 2002). L'algorithme décompose le problème en sous-problèmes en utilisant le nombre de variables. Les sous-populations coévoluent selon les principes d'une collaboration entre elles pour atteindre le front de Pareto.

### 1.6.5 Collaboration entre plusieurs méthodes

L'objectif de la collaboration de plusieurs méthodes d'optimisation est de pouvoir profiter des avantages de chaque méthode. Par exemple, une LS converge rapidement et un EA diversifie la recherche dans l'espace de solutions. L'association des deux méthodes permet d'obtenir une métaheuristique hybride qui á une bonne convergence tout en diversifiant sa recherche. Les hybridations de méthodes d'optimisation sont nombreuses et variées (Blum et al., 2011) et ont de multiples applications.

La TS qui est une amélioration de la LS car elle peut sortir d'un minimum local. Cet algorithme peut aller sur des solutions plus mauvaises que la solution courante et interdit certains mouvements pour éviter un cycle dans la recherche. Pour améliorer l'exploration de la TS, les principes de la SS peuvent être utilisés (Greistorfer, 2003). La Méthode de Nelder-Mead peut aussi utiliser le TS pour éviter de tomber dans un minimum local (Chelouah and Siarry, 2005). Le SA qui est aussi une amélioration de la LS est utilisé pour améliorer les performances d'une métaheuristique avec population. Par exemple, une collaboration entre le PSO et le SA (Behnamian and Fatemi Ghomi, 2010). Le travail en équipe peut exister aussi tel que faire tourner un PSO et un ACO parallèlement et à chaque itération les populations des deux sont réunies pour sélectionner les meilleures solutions (Kiran et al., 2012). D'autres algorithmes hybridés peuvent être simplement l'ajout d'un module tel qu'une archive pour l'AbYSS (Glover, 1998) ou un mécanisme spécifique d'amélioration (Yin et al., 2007).

## 1.7 Conclusion

Dans cet état de l'art, nous avons listé les différentes méthodes d'optimisation : les méthodes exactes pour trouver les solutions optimales, les méthodes approchées avec ou sans population pour résoudre les problèmes NP-difficiles et les hybridations qui permettent d'améliorer les performances des méthodes. Les méthodes d'optimisation utilisent l'espace des objectifs pour orienter leur recherche des meilleures solutions. Les programmations linéaires et par contraintes utilisent des modèles mathématiques pour explorer et trouver la ou les solutions optimales. Le B&B à l'aide de l'évaluation des solutions explorées opère des coupes dans l'espace de recherche. Pour l'ACO, les mémoires sont utilisées pour établir des lois de probabilité sur les valeurs à affecter aux variables. À chaque itération, ces lois de probabilité servent ensuite à construire des solutions. Après évaluation, les mémoires sont mises à jour avec l'aide de la population courante. Lors d'une itération, le PSO modifie aléatoirement chaque particule. Mais les meilleures solutions explorées par l'algorithme et la particule influencent sur cette modification. Dans la LS, le SA et le TS le voisinage de la solution courante est exploré pour tenter d'améliorer l'évaluation. Pour l'hybridation, le FLC modifie la convergence et la diversité de la métaheuristique en fonction de la progression de l'évaluation. Donc, l'exploration de toutes les méthodes d'optimisation est guidée dans l'espace des objectifs.

L'approche proposée dans cette thèse est l'utilisation d'une orientation de la recherche dans l'espace des variables. En connaissant la position des solutions dans l'espace de recherche, des zones intéressantes pour l'exploration sont identifiées. L'exploration des méthodes d'optimisation sera faite en utilisant la position des solutions dans l'espace des objectifs et des variables. Pour cela, une représentation simple de l'espace de solutions par un espace unidimensionnel ordonnant les solutions par rapport à une référence est proposée. Une proposition d'utilisation efficace en découpant la carte unidimensionnelle en zones contenant des solutions avec les mêmes caractéristiques est ensuite abordée pour les méthodes d'optimisation mono-objectif et multiobjectif.

# Chapitre 2

# Espace de solutions représenté dans un espace uni-dimensionnel

## 2.1 Introduction

L'espace des objectifs est utilisé par les méthodes d'optimisation lors de la recherche des meilleures solutions. Dans le but d'améliorer les performances de ces méthodes, cette thèse propose l'utilisation des variables. Pour cela, l'espace de solutions sera représenté par un axe unidimensionnel où chaque solution est associée bijectivement à un numéro. Ce chapitre présente une Méthode de Conversion de l'Espace de recherche (MCE) qui transforme l'espace de solutions en une carte uni-dimensionnelle. Après une conversion en binaire de la solution, la distance de *Hamming* est utilisée pour classer les solutions par rapport à une solution de référence. La carte est découpée en zone, où chacune contient les solutions avec la même distance de *Hamming*. Cette carte permet de situer facilement une solution dans l'espace de recherche. L'objectif est de pouvoir l'utiliser pour guider l'exploration d'une métaheuristique. La qualité de l'exploration des zones par une population est définie par les zones inexplorées ($zi$) et explorées ($ze$). Les $ze$ peuvent être requalifiées comme $znd$ pour celles contenant au moins une des meilleures solutions et/ou $zfe$ pour celles contenant plus que la moyenne des individus présents dans les $ze$. La RL a été retenue pour être utilisée avec la MCE. Généralement, une Phase de Recherche Locale (PRL) est appliquée périodiquement sur les meilleures solutions ; ceci permet d'accélérer la convergence d'une métaheuristique. Avec la MCE, la LS est appliquée sur les solutions appartenant à un certain type de zone $znd$, $zfe$ et $zi$. Des solutions sont créées dans les $zi$ avant de leur appliquer une LS. Le type de LS choisie est le *Simple Hill Climbing* (HC). Le HC améliore successivement une solution en prenant à chaque fois le plus proche voisin améliorant.

Des expérimentations sont faites avec un problème d'ordonnancement NP-difficile : le *Flexible Job Shop Problem* avec l'unique objectif de réduire le maximum des dates de fin de réalisation des lots, le *makespan*. Des méthodes de la littérature sont implémentées pour être ensuite hybridées

et comparées. Une hybridation avec une LS sur les meilleures solutions est aussi comparée avec les algorithmes hybridés par la MCE. Les tests montrent une nette amélioration des algorithmes hybridés par la MCE.

## 2.2 Description du problème

Dans un problème d'optimisation, il existe deux ensembles. Le premier espace est celui des variables du problèmes qui contient les solutions réalisables. Il est généralement multidimensionnel et difficile à manipuler. Chaque solution dans l'espace des variables possède une image dans le second espace, celui des objectifs. Un point dans l'espace des objectifs peut ne correspondre à aucune, une seule ou plusieurs solutions dans l'espace des variables. Dans un problème mono-objectif, l'espace des objectifs comporte un seul axe, par contre pour les problèmes multiobjectifs, il est composé de plusieurs axes. Pour ce cas, il est facile de le réduire à un seul axe avec le classement par front de Pareto. Il est donc aussi très facile de l'utiliser pour orienter la recherche d'une métaheuristique.

L'espace des variables est difficile à manipuler du fait qu'il soit multidimensionnel. Un objectif de la thèse est de proposer une représentation unidimensionnelle de l'espace des variables. Comme dans le cas de l'espace des objectifs, cet axe est facile à utiliser pour orienter l'exploration d'une métaheuristique. En combinant les deux axes représentant chacun l'espace des objectifs et celui des variables, les solutions peuvent être positionnées dans l'espace de recherche en fonction de leur évaluation et de leurs caractéristiques (voir Figure 2.1). Grâce à cette collaboration, un guidage est proposé en combinant les informations dans l'espace des objectifs et celui des variables. Ce guidage propose d'autres axes d'exploration pour améliorer la convergence et la diversification d'une métaheuristique.

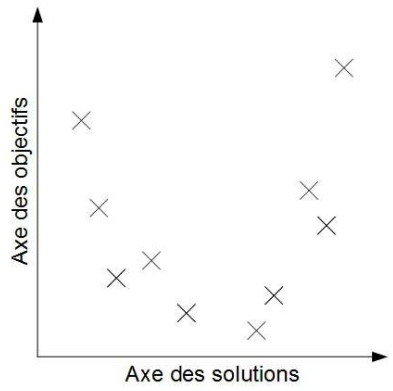


FIGURE 2.1 – Représentation des espaces d'un problème d'optimisation

Comme pour l'axe des objectifs, celui des variables a un point d'origine ou une solution de référence. Le choix de ce point d'origine a son importance car il détermine la répartition des solutions

sur l'axe des variables. Les solutions réalisables doivent être bien réparties sur l'axe des solutions. Par exemple, subdiviser l'axe en zones est un moyen simple pour se localiser. Les solutions à l'intérieur d'une zone ont des caractéristiques communes. Dans ce cas, pour identifier avec pertinence des groupes de solutions, il est préférable d'avoir le maximum de zones avec des solutions réalisables.

## 2.3 Méthode de Conversion de l'Espace de recherche

La Méthode de Conversion de l'Espace de recherche (MCE) fait correspondre bijectivement chaque point dans l'espace des variables à un autre dans espace uni-dimensionnel (voir Figure 2.2).

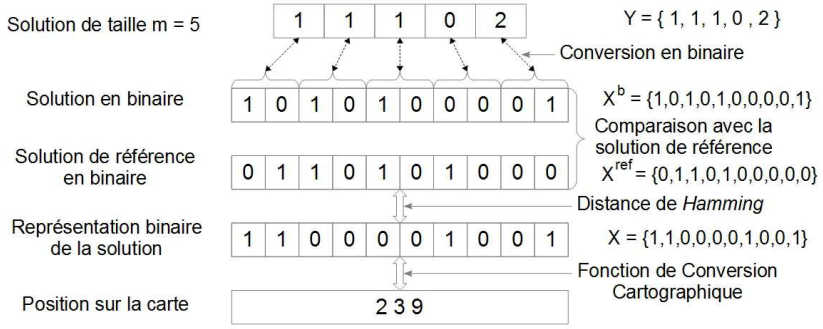


FIGURE 2.2 – Méthode de Conversion Cartographique (MCE)

Les notations utilisées pour la MCE sont les suivantes :

— $Y = \{y_1, ..., y_m\}$ : solution de taille $m$, $y_i \in \mathbb{N}$, $\forall i \in \{1, ..., m\}$.
— $X^b = \{x_1^b, ..., x_n^b\}$ : conversion binaire de la solution $Y$, $x_i^b \in \{0, 1\}$, $\forall i \in \{1, ..., n\}$, avec $n \geq m$.
— $X^{ref} = \{x_1^{ref}, ..., x_n^{ref}\}$ : solution de référence en binaire.
— $X = \{x_1, ..., x_n\}$ : représentation binaire de la solution $X^b$, $x_i = d_H(x_i^b, x_i^{ref})$ $\forall i \in \{1, ..., n\}$.
— $d_H(X) = \sum_{i=1}^{n} x_i$ : distance de *Hamming* $X^b$ par rapport à $X^{ref}$.
— $e(X, k) = \sum_{i=1}^{k} x_i$.
— $a$ : position de $X$ sur la carte.

La MCE transforme un espace de solution multidimensionnel en une carte uni-dimensionnelle, en quatre étapes :

1. Chaque solution $Y$ est convertie en binaire, $Y \longrightarrow X^b$.
2. Comparaison de $X^b$ avec $X^{ref}$, $X = d_H(x_i^b, x_i^{ref})$, $\forall i \in \{1, ..., n\}$.
3. $X = \{x_1, ..., x_n\}$ est la représentation binaire de la solution $Y$ sur la carte.
4. Conversion décimale de $X$ utilisant la Fonction de Conversion Cartographique (FCC) $f(X)$ (voir équation (2.1) et (2.2)).

La Fonction de Conversion Cartographique $f(X)$ (FCC) est définie comme suit :

$$\begin{array}{lll} D = \{0,1\}^n & \longrightarrow & I \in \mathbb{N} \\ f(X = \{x_1, ..., x_n\}) & \longrightarrow & a \end{array} \tag{2.1}$$

avec

$$f(X) \quad = \quad \sum_{i=0}^{d_H(X)-1} C_n^i + \sum_{i=1}^{n} \left[ C_{i-1}^{e(X,i)-1} \cdot (1 - x_i) \right] \tag{2.2}$$

**Exemple 1** (Calcul de $f(X)$)**.** *Dans l'exemple de la Figure 2.2, la solution est :* $Y = \{11102\}$, $m = 5$. *La solution de référence est une solution faisable créée aléatoirement :* $Y^{ref} = \{21110\}$. *La solution en binaire est :* $X^b = \{1010100001\}$, $n = 10$. *La solution de référence en binaire est :* $X^{ref} = \{0110101000\}$. *En comparant la solution* $X^b$ *avec la référence* $X^{ref}$ *avec la distance de Hamming, la représentation binaire obtenue est* $X = \{1100001001\}$. *La distance de Hamming de* $X$ *est* $d_H(X) = \sum_{i=1}^{n} x_i = 4$. *La position de la solution* $X$ *est donc dans la zone 4. Dans cet exemple, il y a* $n = 10$ *zones.*

*La position sur la carte est calculée comme suit (voir équation (2.2)) :*

$$\begin{array}{lll} f(X) & = & \sum_{i=0}^{d_H(X)-1} C_n^i + \sum_{i=1}^{n} \left[ C_{i-1}^{e(X,i)-1} \cdot (1 - x_i) \right] \\ & = & \sum_{i=0}^{3} C_{10}^i + C_0^0 \cdot (1-1) + C_1^0 \cdot (1-1) + C_2^1 \cdot (1-0) + \\ & & C_3^1 \cdot (1-0) + C_4^1 \cdot (1-0) + C_5^1 \cdot (1-0) + C_6^1 \cdot (1-1) + \\ & & C_7^2 \cdot (1-0) + C_8^2 \cdot (1-0) + C_9^2 \cdot (1-1) \\ & = & C_{10}^0 + C_{10}^1 + C_{10}^2 + C_{10}^3 + C_2^1 + C_3^1 + C_4^1 + C_5^1 + C_7^2 + C_8^2 \\ & = & 1 + 10 + 45 + 120 + 2 + 3 + 4 + 5 + 21 + 28 \\ a & = & 239 \end{array}$$

**Propriété 1.** *Avec la formule de Pascal, les propriétés suivantes sont vérifiées :* $\forall X = \{x_1, ..., x_n\}$ *avec* $x_i \in \{0,1\}$, $\forall i \in \{1, ..., n\}$ *:*

$$C_n^{d_H(X)} \quad > \quad \sum_{i=1}^{n} \left[ C_{i-1}^{e(X,i)-1} \cdot (1 - x_i) \right] \tag{2.3}$$

*Démonstration.* Pour simplifier les expressions et sans perte de généralités, nous écrivons $d_H(X)$ par $d_H$.
Selon la formule de Pascal, $\forall n \in \mathbb{N}$ and $\forall d_H \in \{1, ..., n\}$ :

$$\begin{array}{lll} C_n^{d_H} & = & C_{n-1}^{d_H} + C_{n-1}^{d_H-1} \quad = \quad C_{n-1}^{d_H-1} + \sum_{i=d_H-1}^{n-2} C_i^{d_H-1} \\ & = & \sum_{i=d_H-1}^{n-1} C_i^{d_H-1} \quad > \quad \sum_{i=d_H+1}^{n} C_{i-1}^{d_H-1} \end{array} \tag{2.4}$$

En utilisant (voir équation (2.4)) et $\forall k \in \{d_H + 1, ..., n\}$ :

$$C_n^{d_H} > \sum_{i=d_H+1}^{n} C_{i-1}^{d_H-1} \geq C_{k-1}^{d_H-1} + \sum_{i=k+1}^{n} C_{i-1}^{d_H-1}$$

En appliquant encore (voir équation (2.4)) pour $C_{k-1}^{d_H-1}$, nous obtenons :

$$C_n^{d_H} > \sum_{i=d_H-1}^{k-1} C_{i-1}^{d_H-2} + \sum_{i=k+1}^{n} C_{i-1}^{d_H-1} \tag{2.5}$$

L'inégalité (voir équation (2.5)) peut être généralisée comme suit :
Soit $k_j \in \{1, ..., n-1\} \backslash k_j < k_{j+1}$, $\forall j \in \{1, ..., d_H - 1\}$.

$$C_n^{d_H} > \sum_{j=1}^{d_H-1} \sum_{i=k_j+1}^{k_{j+1}-1} C_{i-1}^{j-1} + \sum_{i=k_{d_H}+1}^{n} C_{i-1}^{d_H-1} \tag{2.6}$$

Soit $X = \{x_i\}_{1 \leq i \leq n}$ les termes correspondants aux positions $\{k_j\}_{1 \leq j \leq d_H}$,
où :

$$x_i = \begin{cases} 1 & si\ i \in \{k_j\}_{1 \leq j \leq d_H} \\ 0 & sinon \end{cases}$$

En plus, $e(X, i) = \sum_{l=1}^{i} x_l = j$, $\forall i \in \{k_j, ..., k_{j+1} - 1\}$.
Finalement, l'inégalité (voir équation (2.6)) peut être réécrite ainsi :

$$C_n^{d_H} > \sum_{j=1}^{d_H-1} \sum_{i=k_j+1}^{k_{j+1}-1} \left[ C_{i-1}^{e(X,i)-1} \cdot (1 - x_i) \right] + \sum_{i=k_{d_H}+1}^{n} \left[ C_{i-1}^{e(X,i)-1} \cdot (1 - x_i) \right] \tag{2.7}$$

Grâce à $\sum_{i=1}^{k_1} \left[ C_{i-1}^{e(X,i)-1} \cdot (1 - x_i) \right]$ n'est pas défini sur $i \in \{1, ..., k_1 - 1\}$ parce que $e(X, i) - 1 < 0$.
L'inégalité (voir équation (2.7)) peut être réécrite comme suit : $\forall X = \{x_1, ..., x_n\}$ avec $x_i \in \{0, 1\}$, $\forall i \in \{1, ..., n\}$ :

$$C_n^{d_H(X)} > \sum_{i=1}^{n} \left[ C_{i-1}^{e(X,i)-1} \cdot (1 - x_i) \right]$$

□

**Propriété 2.** *La MCE est injective.*

*Démonstration.* Soit deux solutions distinctes $X^1$ et $X^2 \setminus X^1 \neq X^2; X^1 = \{x_1^1, ..., x_n^1\}$ et $X^2 = \{x_1^2, ..., x_n^2\}$. Deux cas existent : $d_H(X^1) = d_H(X^2)$ et $d_H(X^1) \neq d_H(X^2)$.

**Cas 1 :** Si $d_H(X^1) = d_H(X^2)$

$\exists\ k \in \{1, ..., n\} \setminus$

$$x_i^1 \begin{cases} = x_i^2 & si\ i > k \\ \neq x_i^2 & si\ i = k \end{cases}$$

Soit $\Delta f = f(X^1) - f(X^2)$.

$$\begin{aligned} \Delta f &= \sum_{i=1}^{n} \left[ C_{i-1}^{e(X^1,i)-1} \cdot (1 - x_i^1) \right] - \sum_{i=1}^{n} \left[ C_{i-1}^{e(X^2,i)-1} \cdot (1 - x_i^2) \right] \\ &= \sum_{i=1}^{k} \left[ C_{i-1}^{e(X^1,i)-1} \cdot (1 - x_i^1) \right] - \sum_{i=1}^{k} \left[ C_{i-1}^{e(X^2,i)-1} \cdot (1 - x_i^2) \right] \end{aligned} \tag{2.8}$$

Deux sous-cas sont considérés :

**Sous-cas 1.1 :** Si $x_k^1 = 0$ et $x_k^2 = 1$.

$\Delta f$ peut être réécrit :

$$\begin{aligned} \Delta f &= C_{k-1}^{e(X^1,k)-1} + \sum_{i=1}^{k-1} \left[ C_{i-1}^{e(X^1,i)-1} \cdot (1 - x_i^1) \right] \\ &\quad - \sum_{i=1}^{k-1} \left[ C_{i-1}^{e(X^2,i)-1} \cdot (1 - x_i^2) \right] \end{aligned}$$

Et, selon l'inégalité (voir équation (2.3)) :

$$C_{k-1}^{e(X^1,k)-1} = C_{k-1}^{e(X^2,k)-1} > \sum_{i=1}^{k-1} \left[ C_{i-1}^{e(X^2,i)-1} \cdot (1 - x_i^2) \right] \tag{2.9}$$

Donc

$$\Delta f > \sum_{i=1}^{k-1} \left[ C_{i-1}^{e(X^1,i)-1} \cdot (1 - x_i^1) \right] \geq 0 \tag{2.10}$$

Alors : si $X^1 \neq X^2$, $f(X^1) \neq f(X^2)$

**Sous-cas 1.2 :** Si $x_k^1 = 1$ et $x_k^2 = 0$, alors :

$$\begin{aligned} \Delta f &= -C_{k-1}^{e(X^2,k)-1} + \sum_{i=1}^{k-1} \left[ C_{i-1}^{e(X^1,i)-1} \cdot (1 - x_i^1) \right] \\ &\quad - \sum_{i=1}^{k-1} \left[ C_{i-1}^{e(X^2,i)-1} \cdot (1 - x_i^2) \right] \end{aligned}$$

Et selon l'équation (voir équation (2.3)) :

$$-C_{k-1}^{e(X^2,k)-1} = -C_{k-1}^{e(X^1,k)-1} < -\sum_{i=1}^{k-1} \left[ C_{i-1}^{e(X^1,i)-1} \cdot (1 - x_i^1) \right] \tag{2.11}$$

Donc :

$$\Delta f < -\sum_{i=1}^{k-1} \left[ C_{i-1}^{e(X^2,i)-1} \cdot (1 - x_i^2) \right] \leq 0 \tag{2.12}$$

Finalement, avec (voir équations (2.10) et (2.12)) : $\Delta f \neq 0$, donc si $X^1 \neq X^2$, $f(X^1) \neq f(X^2)$.

**Cas 2 :** Si $d_H(X^1) \neq d_H(X^2)$ : deux sous-cas sont à considérer :

**Sous-cas 2.1 :** Si $d_H(X^1) < d_H(X^2)$ alors, d'après la formule de Pascal (voir l'inégalité (2.3)) :

$$f(X^1) = \sum_{i=0}^{d_H(X^1)-1} C_n^i + \sum_{i=1}^{n} \left[ C_{i-1}^{e(X^1,i)-1} \cdot (1 - x_i^1) \right]$$

$$\begin{aligned} &< \sum_{i=0}^{d_H(X^1)-1} C_n^i + C_n^{d_H(X^1)} \\ &< \sum_{i=0}^{d_H(X^1)} C_n^i \\ &< \sum_{i=0}^{d_H(X^2)-1} C_n^i + \sum_{i=1}^{n} \left[ C_{i-1}^{e(X^2,i)-1} \cdot (1 - x_i^2) \right] \quad = \quad f(X_2) \end{aligned} \tag{2.13}$$

Donc, si $d_H(X^1) < d_H(X^2)$ alors $f(X^1) < f(X^2)$.

**Sous-cas 2.2 :** Si $d_H(X_1) > d_H(X^2)$ :

$$f(X^2) = \sum_{i=0}^{d_H(X^2)-1} C_n^i + \sum_{i=1}^{n} \left[ C_{i-1}^{e(X^2,i)-1} \cdot (1 - x_i^2) \right]$$

$$\begin{aligned} &< \sum_{i=0}^{d_H(X^2)-1} C_n^i + C_n^{d_H(X^2)} \\ &< \sum_{i=0}^{d_H(X^2)} C_n^i \\ &< \sum_{i=0}^{d_H(X^1)-1} C_n^i + \sum_{i=1}^{n} \left[ C_{i-1}^{e(X^1,i)-1} \cdot (1 - x_i^1) \right] \quad = \quad f(X_1) \end{aligned} \tag{2.14}$$

D'après (voir équations 2.13) et (2.14) : Si $d_H(X^1) \neq d_H(X^2)$, $f(X^1) \neq f(X^2)$.

Finalement, selon les équations (2.10), (2.12), (2.13) et (2.14) la fonction $f(X)$ est injective.

□

**Propriété 3.** *La MCE est surjective.*

*Démonstration.* Pour prouver que la MCE est surjective, il suffit que $|D| = |I| = 2^n$. Soit la solution $X^{lb}$ définie par :

$$x_i^{lb} = 0, \quad \forall i \in \{1, ..., n\}$$

et $X^{lh}$ est définie par :

$$x_i^{lh} = 1, \quad \forall i \in \{1, ..., n\}$$

$$f(X^{lb}) = \sum_{i=0}^{d_H(X^{lb})-1} C_n^i + \sum_{i=1}^{n} \left[ C_{i-1}^{e(X^{lb},i)-1} \cdot (1 - x_i^{lb}) \right] = 0 \tag{2.15}$$

Alors $f(X) \geq 0$, $\forall X \in D$, donc la borne inférieure de $I$ est 0.

$$\begin{aligned} f(X^{lh}) &= \sum_{i=0}^{d_H(X^{lh})-1} C_n^i + \sum_{i=1}^{n} \left[ C_{i-1}^{e(X^{lh},i)-1} \cdot (1 - x_i^{lh}) \right] \\ &= \sum_{i=0}^{n-1} C_n^i + \sum_{i=1}^{n} \left[ C_{i-1}^{i-1} \cdot (1-1) \right] \\ &= \sum_{i=0}^{n-1} C_n^i \\ &= \sum_{i=0}^{n} C_n^i - 1 \end{aligned} \tag{2.16}$$

En plus, $\forall X \in D$ :

$$\begin{aligned} f(X) &= \sum_{i=0}^{d_H(X)-1} C_n^i + \sum_{i=1}^{n} \left[ C_{i-1}^{e(X,i)-1} \cdot (1 - x_i) \right] \\ &< \sum_{i=0}^{d_H(X)-1} C_n^i + C_n^{d_H(X)} \\ &< \sum_{i=0}^{d_H(X)} C_n^i \\ &\leq \sum_{i=0}^{n} C_n^i - 1 = f(X^{lh}) \end{aligned} \tag{2.17}$$

En conséquence de l'injectivité de $f(X)$ et $|D| = 2^n$, alors $|I| = \sum_{i=0}^{n} C_n^i - 1 + 1 = 2^n$. Nous pouvons conclure aussi que $X^{lh}$ est la borne supérieure de $I$.
Finalement, nous pouvons en déduire que la MCE est surjective.

□

**Propriété 4.** *La MCE est bijective.*

*Démonstration.* Selon les propriétés 1 et 2, nous pouvons conclure que la MCE est bijective.

□

## 2.4 Répartition des solutions sur la carte uni-dimensionnelle

Pour l'exemple de Figure 2.2, le nombre de bits est $n = 10$ et le nombre de solutions est $2^{10} = 1024$. Les solutions sont ordonnées en fonction de leur distance de *Hamming*, plus cette dernière est grande, plus la position est éloignée sur la carte de la solution de référence (voir Figure 2.3). La distance de *Hamming* est utilisée pour découper la carte en zone. Le nombre de solutions dans chaque zone $i = 1, ..., n$ est égale à $C_n^i - C_n^{i-1}$. La Figure 2.3 montre la répartition des solutions sur la carte en fonction de la distance de *Hamming* comparée à la solution de référence.

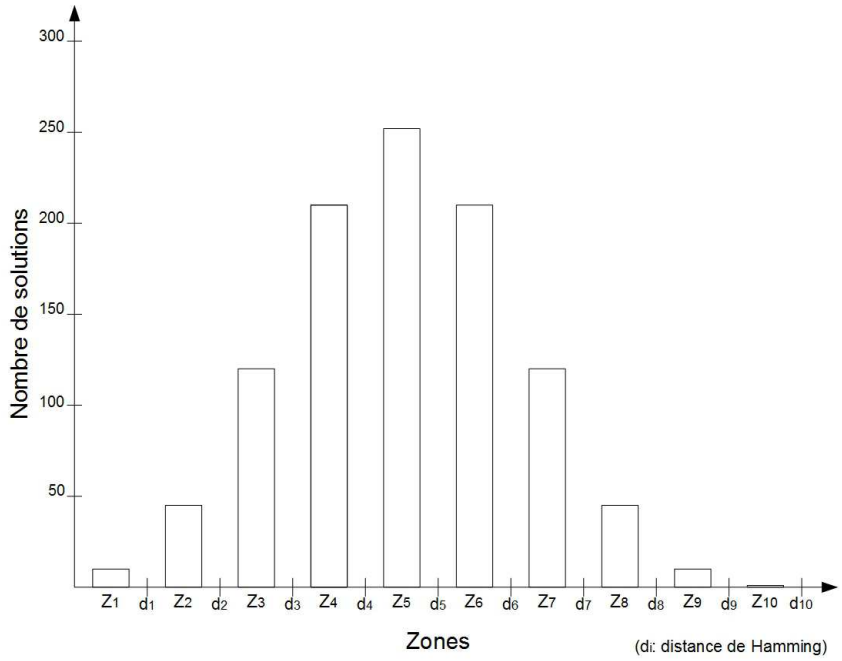


FIGURE 2.3 – Répartition des solutions sur la carte uni-dimensionnelle (exemple de la Figure 2.2)

## 2.5 Choix de la solution de référence

Plusieurs types de solutions de référence sont possibles. La plus évidente est la solution zéro ($\{X_i = 0\}_{1 \geq i \geq n}$), mais il faut faire attention à la répartition des solutions faisables sur la carte. Un mauvais choix pour la solution de référence peut entraîner la concentration des solutions faisables dans une ou un petit nombre de zones, annulant de fait l'intérêt de la MCE. La Figure 2.4 présente une répartition des solutions explorées concentrées dans quelques zones, ce qui empêche d'identifier efficacement les groupes de solutions intéressantes. Une meilleure répartition des solutions explorées dans de nombreuses zones (voir Figure 2.5) permet de mieux différencier les ensembles de solutions.

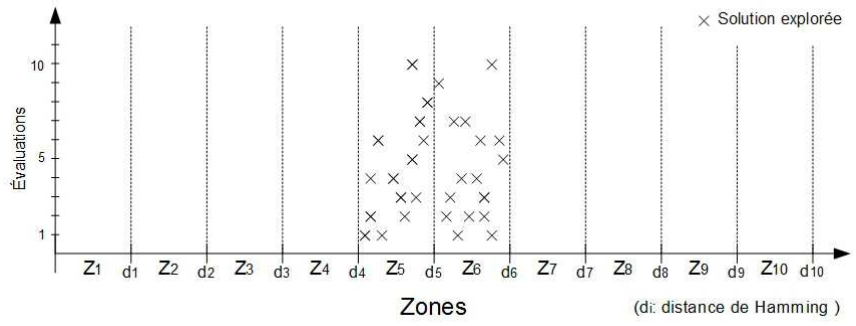


FIGURE 2.4 – Concentration des solutions explorées dans deux zones

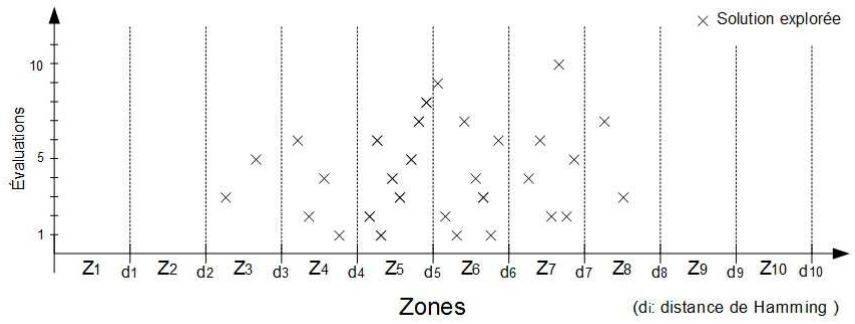


FIGURE 2.5 – Répartition sur de nombreuses zones

## 2.6 Orientation sur la carte uni-dimensionnelle

Sur la carte, les solutions sont classées en fonction de leur différence par rapport à une solution de référence. La solution doit fournir une bonne répartition des solutions faisables sur la carte unidimensionnelle.

Les solutions infaisables sont à exclure, car elles peuvent être trop éloignées de l'espace de solutions faisables. Dans ce cas les solutions faisables se localiseront dans un petit espace de la carte unidimensionnelle supprimant ainsi l'intérêt de la carte unidimensionnelle.

Les solutions sont converties en binaire pour être classées en fonction de la distance de *Hamming* par rapport à la référence. Les solutions avec la même distance de *Hamming* sont regroupées à l'intérieur d'une zone. Dans une zone, les solutions sont classées selon la position des bits à 1 dans leur représentation binaire.

Lors de l'exploration de l'espace de solutions par une métaheuristique, une zone est évaluée en fonction de la qualité et du nombre de solutions explorées à l'intérieur.

Les zones sont classées en deux catégories : les $zi$ ne contenant aucune solution explorée et les $ze$ contenant au moins une solution explorée par la population courante. Les $ze$ peuvent être qualifiées de $znd$ si elle contient au moins une des meilleures solutions et/ou de $zfe$ si elle contient plus de solutions explorées que la moyenne des $ze$. Une $ze$ peut être à la fois explorée $znd$ et $zfe$.

— $zi$ : Une zone inexplorée ne contient aucun individu de la population courante.
— $ze$ : Une zone explorée inclue au moins un individu de la population courante, ce type peut être classifié dans deux catégories :

- $znd$ : Une zone qui contient au moins un des meilleurs individus de la population.
- $zfe$ : Une zone fortement explorée contient un nombre d'individus supérieur à la moyenne d'individus présents dans les $ze$s.

Dans l'exemple de la Figure 2.6, le problème est multiobjectif. La taille de la population est 30 avec 4 solutions non-dominées et le nombre de zones est 10. Les zones 1, 2, 9 et 10 sont des $zi$ et

les *ze* sont : 3, 4, 5, 6, 7 et 8. La moyenne du nombre d'individus dans les *ze* est égal à $30/6 = 5$ (i.e. $miz = 5$). Et par suite, les *zfe* sont 5, 6 et 7 et les *znd* sont 4, 5 et 6.

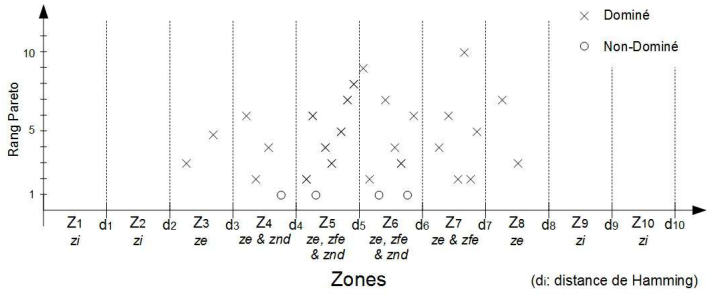


FIGURE 2.6 – Exemple de distribution de la population sur la carte uni-dimensionnelle (exemple de la Figure 2.2 et 2.3)

## 2.7 Utilisation de la MCE

Pour les métaheuristiques, l'équilibre entre la convergence, pour trouver rapidement les meilleures solutions, et la diversification, pour éviter les minimums locaux, est important pour une bonne exploration de l'espace de solutions. Ce principe est utilisé pour l'orientation de la recherche de solutions sur la carte unidimensionnelle. L'exploration de la méthode d'optimisation peut être améliorée par deux hybridations : en convergence, notée "MCE-Convergence", et/ou en diversité, notée "MCE-Diversité". La recherche locale est utilisée avec la MCE. Le principe est d'appliquer la recherche locale sur les solutions des zones intéressantes. Pour la MCE-Convergence, les zones identifiées sont les *znd* ou les $zfe$. Les *znd* contiennent les meilleures solutions et celles avec la même distance de *Hamming* de la population. Les *zle* ont un nombre de solutions explorées supérieures à la moyenne dans les *ze*s. Pour la MCE-Diversité, les zones identifiées sont les $zi$. En créant des solutions dans ces zones de nouvelles caractéristiques sont introduites dans la population. Une LS est appliquée sur ces nouvelles solutions.

— MCE-Convergence
  - Identifier les $ze$
  - Identifier les *znd* et $zfe$
  - Appliquer MCE1 ou MCE2

— MCE-Diversité
  - Identifier les $zi$
  - Appliquer MCE3

Où MCE1, MCE2 et MCE3 sont définies comme suit :

— MCE1 : Une recherche locale est appliquée sur les solutions dans les $ndz$.
— MCE2 : Une recherche locale est appliquée sur les solutions dans les $zfe$.
— MCE3 : Des nouvelles solutions sont créées dans les $zi$ (voir Algorithme 2.2 et une recherche locale est appliquée sur chacune d'elles.

MCE4 est définie comme l'application de MCE1 et MCE3 simultanément. MCE5 est la combinaison de MCE2 et MCE3.

### 2.7.1 Notations

Les notations utilisées sont les suivantes :

— $si$ : Solution initiale.
— $sc$ : Solution courante.
— $ns$ : Nouvelle solution.
— $na$ : Nombre d'améliorations.
— $li$ : Nombre maximum d'améliorations.
— $ne$ : Nombre d'essais.
— $le$ : Limite du nombre d'essais.
— $eiz$ : Ensemble des zones inexplorées.
— $zns$ : Zone de la nouvelle solution.

### 2.7.2 Indicateur MCE

Pour décider de l'application d'une phase MCE, l'indicateur MCE ($Ind-MCE$) est effectué périodiquement. La progression de la meilleure note pour le mono-objectif ou du front de Pareto évalué par la métrique-C (Ziztler et al., 2003) pour le multiobjectif, est utilisé pour évaluer la convergence. Si la progression est quasi-nulle une phase MCE est appliquée sur la population courante.

Les équations 2.18 et 2.19 présentent le calcul de l'Ind-MCE pour le mono-objectif et le multiobjectif :

$$Ind - Mce = MS_{i-T} - MS_i \tag{2.18}$$

$$Ind - Mce = C(FPC_i, FPC_{i-T}) \tag{2.19}$$

Avec $MS_i$ la Meilleure Solution et $FPC_i$ le Front Pareto Courant à l'itération $i$ et $T$ la période d'application d'une PRL.

### 2.7.3 Description de la Recherche Locale

La LS utilisée est le *Simple Hill Climbing* (HC). Le HC améliore plusieurs fois de suite une solution, jusqu'à atteindre son critère d'arrêt. L'algorithme 2.1 présente le HC.

**Algorithme 2.1** Recherche locale : *Simple Hill Climbing*

**Paramètre** $si$ : Solution initiale

**Paramètre** $le$ : Nombre limite d'essais

1: $sc \leftarrow si$
2: $ne \leftarrow 1$
3: **Tant que** $ne \leq le$ **Faire**
4: $ns \leftarrow$ voisinage($sc$)
5: Évaluation($ns$)
6: $ne \leftarrow ne + 1$
7: **Si** $ns > sc$ **Alors**
8: $sc \leftarrow ns$
9: $ne \leftarrow 0$
10: **Fin Si**
11: **Fin Tant que**
12: **Retourne** $sc$

La solution initiale $si$ reste dans la population et la nouvelle solution $ns$ est ajoutée. Cette variante permet de garder une bonne diversité dans la population. Deux politiques de collaborations entre la LS et la MCE sont possibles. La première, la LS reste dans la zone de la solution $si$, mais ceci demande plus de calcul et limite les améliorations possibles. La seconde autorise la LS à sortir de la zone de $si$. Cette dernière a été choisie car elle fournit plus d'opportunités d'amélioration et demande moins de temps de calcul.

### 2.7.4 Création de solutions dans les zones inexplorées

La fonction perturbation est une exploration itérative du voisinage de la solution $si$ (voir l'algorithme 2.2). Si une solution du voisinage est dans une zone inexplorée, l'algorithme retourne cette solution. Si l'algorithme atteint la limite $le$, il arrête son exploration du voisinage.

## 2.8 Application sur un problème mono-objectif NP-Difficile

Le problème choisi pour expérimenter la MCE est un problème d'ordonnancement : le *Flexible Job Shop Problem* (FJSP) avec pour objectif la minimisation du *makespan*. Dans cette partie, un état de l'art présente quelques méthodes d'optimisation pour résoudre le FJSP, ce problème est ensuite

**Algorithme 2.2** Création de solution dans les zones inexplorées

**Paramètre** $si$ : Solution initiale

**Paramètre** $ne$ : Nombre limite d'essais

**Paramètre** $eiz$ : Ensemble des zones inexplorées

1: $ne \leftarrow 1$
2: **Tant que** $ne \leq le$ **Faire**
3: $ns \leftarrow$ Perturbation($si$)
4: **Si** $nsz \in eiz$ **Alors**
5: **Retourne** $ns$
6: **Fin Si**
7: $ne \leftarrow ne + 1$
8: **Fin Tant que**

défini dans la section suivante. Les sections suivantes contiennent la description des métaheuristiques pour résoudre le FJSP, la présentation des expérimentations et la comparaison des résultats.

### 2.8.1 État de l'art du *Flexible Job Shop Problem* avec l'objectif minimisation du *makespan*

Cette section donne un court état de l'art sur de récentes méthodes pour résoudre le *Flexible Job Shop Problem* (FJSP) avec l'objectif classique de minimiser le maximum des *completion times* des lots (*makespan*).

Le GA est l'une des métaheuristiques les plus utilisées pour résoudre le FJSP (Pezzella et al., 2008). La population initiale est créée par des heuristiques utilisées pour l'affectation et le séquençage des opérations dans les problèmes d'opérations (Kacem et al., 2002a). Deux règles d'affection des opérations sont utilisées : la première choisit systématiquement la machine avec le plus petit temps de réalisation pour l'opération et la seconde la choisit aléatoirement. Pour l'ordre de traitement des opérations sur les machines, trois règles sont utilisées : le choix aléatoire d'une opération, la *Most Work Remaining* (MWR) et la *Most number of Operations Remaining* (MOR). À chaque itération, les règles MWR et MOR ont deux politiques pour choisir le lot de l'opération à affecter. La MWR ordonnance le lot avec le plus de temps avant la fin de sa réalisation. La MOR choisit le lot auquel il reste le plus d'opération à affecter.

Ce GA teste trois types de sélection pour les parents pour la reproduction : le tournoi binaire, le choix aléatoire et le rang linéaire. Dans la sélection par rang linéaire, chaque individu obtient un rang $r_i \in \{1, ..., N\}$ après l'évaluation et le tri. Le meilleur individu a le rang $N$ et le pire le rang 1. Alors la probabilité de chaque individu est déterminée par la formule suivante :

$$p_i = \frac{2r_i}{N(N+1)} \quad i = 1, ..., N$$

où $i$ correspond à l'identifiant du $i^{eme}$ individu.

Pour la reproduction, les opérateurs *Procedence Preserving Order-based crossover* (POX) (Kacem et al., 2002a) et *Precedence Preserving Shift mutation* (PPS) (Kyung-Mi et al., 1998) sont utilisés pour l'ordonnancement des opérations sur les machines. Pour l'affectation des opérations aux machines, le croisement en un point est choisi. Deux types de mutation sont essayés, la plus simple est de changer l'affectation d'une opération et la seconde, dite intelligente, une opération est sélectionnée sur la machine la plus chargée pour être réaffecter à une autre.

Le FSJP est résolu par l'AIS, en autre le *Artificial Immune System* (AIA) (Bagheri et al., 2010). L'AIA utilise la même stratégie que le GA (Pezzella et al., 2008) pour générer sa population initiale (Kacem et al., 2002a). Les individus avec une meilleure affinité ont plus de chance d'être sélectionnés pour le clonage. L'hypermutation, similaire au GA (Pezzella et al., 2008), est une mutation intelligente qui enlève une opération de la machine la plus chargée et le PPS (Kyung-Mi et al., 1998) pour la séquence des opérations.

Le *Knowledge-Based Ant Colony Optimization* (KBACO) (Xing et al., 2010) est une adaptation de l'ACO pour résoudre le FJSP. Le KBACO est une combinaison entre l'ACO et un modèle d'apprentissage. Le *KnowledgeBased Heuristic Searching Architecture* (KBSHA) est divisé en deux modules. Le KBSHA traite les informations venant de l'heuristique pour mettre à jour sa mémoire. Ces connaissances sont ensuite traitées pour guider l'heuristique dans sa recherche de solution.

Le PSO est une métaheuristique qui s'adapte aussi pour le FSJP (Girish and Jawahar, 2009) et (Xia and Wu, 2005). La population initiale est générée pour avoir un bon équilibre entre qualité et diversité. Le déplacement des particules dans l'espace des solutions se fait sous l'influence de la meilleure solution visitée par la particule et de celle visitée par la population. Le calcul de la vitesse $v_i(t)$ et de la position $x_i(t)$ est classique pour un PSO :

$$\begin{aligned} v_i(t+1) &= w * v_i(t) + c_1[x_i^*(t) - x_i(t)] + c_2 * [x_g^*(t) - x_i(t)] \\ x_i(t+1) &= x_i(t) + v_i(t+1) \end{aligned}$$

où $w$ est l'inertie de la particule, $c_1$ et $c_2$ les poids qui déterminent l'influence de la meilleure solution locale $x_i^*(t)$ et globale $x_g^*(t)$.

Le FJSP avec l'objectif de minimiser le *makespan* est résolu efficacement par l'ABC (Wang et al., 2012b). Cet ABC propose une recherche locale utilisant le chemin critique d'une solution qui améliore ainsi l'intensification de la recherche. Les opérations critiques sont identifiées et déplacées pour améliorer la qualité de la solution.

Le FJSP peut être traité par le TS : le *Tabu Search algorithm with Fast Public Critical Block neighborhood structure* (TSPCB) (Li et al., 2011b). Dans cet article, l'auteur propose un TS utilisant une structure de voisinage pour le FJSP. Pour créer les solutions initiales, le TSPCB utilise un mélange de quatre règles pour l'affectation aux machines et l'ordonnancement des opérations. La première règle consiste à créer des solutions purement aléatoirement. La seconde est la règle *operation minimum processing time* (Pezzella et al., 2008) où lors de la construction de la solution c'est l'opération avec le moins de temps de réalisation qui est prioritaire pour l'affectation. La troisième est la *local minimum processing time*, pour chaque lot l'opération avec le moins de temps de réalisation est affectée à une machine. La dernière la *global minimum processing time* (Pezzella et al., 2008) où l'opération dont le lot qui à actuellement le moins de temps de travail a la priorité pour être affectée à une machine. La définition du voisinage rend la recherche locale dans les affectations aux machines efficace. Trois approches pour le voisinage sont proposées : (i) la première où une opération est sélectionnée aléatoirement pour être réaffectée, (ii) la deuxième consiste à prendre une opération sur la machine la plus occupée pour l'affecter sur une autre machine et (iii) la troisième consiste à trouver la machine qui termine le plus tard et identifier dessus une opération critique pour la réaffecter à une autre machine. Le voisinage pour l'ordonnancement est défini par un déplacement d'une opération ou d'un échange de places de deux opérations dans l'ordre de traitement. Les calculs sont simples et rapides ce qui donne de la rapidité à la recherche locale.

### 2.8.2 Définition du problème

Le problème de référence pour appliquer la MCE est le *Flexible Job Shop Problem* (Pezzella et al., 2008). Le FJSP consiste à ordonnancer $n$ lots, notés $J$, $J = \{J_1, ..., J_n\}$ sur $m$ machine $M = \{M_1, ..., M_m\}$. Chaque lot $J_i$ est composé d'un suite ordonnée d'opérations de $h_i$ opérations, $\{O_{i,1}, ..., O_{i,h_i}\}$. Chaque opération $O_{ij}$ possède un ensemble de machines qualifiées $M_{ij}$ pour les réaliser. Le FJSP est composé de deux sous-problèmes. Le premier est l'affectation des opérations à une machine. Le deuxième est d'ordonner les opérations assignées aux machines, comme dans un *Job Shop Problem* (JSP). Le FJSP est NP-Difficile avec l'objectif de minimiser le maximum des dates de fin de réalisation des lots le *makespan* ($C_{max}$) (Garey et al., 1976).

Le FJSP est considéré en mono-objectif avec la minimisation du *makespan*.

La fonction objectif est :

$$\min C_{max} \tag{2.20}$$

Les hypothèses suivantes sont considérées :

— Les opérations sur les lots sont fixées dans un ordre prédéfini.
— Chaque lot a la même priorité.
— Les lots sont prés à l'instant 0 et les machine commence à l'instant 0.
— Les temps de *setup* et de déplacement sont négligeables comparés au temps de réalisation.
— Les machines sont indépendantes les unes des autres.

— Les lots sont indépendants les uns des autres.
— Une machine n'effectue qu'une seule opération à la fois.
— Un lot ne peut avoir qu'une seule solution en réalisation à la fois.
— Les opérations ne peuvent pas être divisées.

Le tableau 2.1 présente un exemple d'instance du FJSP. Elle est constituée de 3 lots chacun avec une suite de 3 opérations qui doivent être réalisées sur 3 machines.

TABLEAU 2.1 – Exemple d'instance (3/3/3)

| | Lot 1 | | | Lot 2 | | | Lot 3 | | |
|---|---|---|---|---|---|---|---|---|---|
| | $O_{11}$ | $O_{12}$ | $O_{13}$ | $O_{21}$ | $O_{22}$ | $O_{23}$ | $O_{31}$ | $O_{32}$ | $O_{33}$ |
| Temps $M_1$ | 1 | - | - | - | 3 | 3 | 4 | - | - |
| Temps $M_2$ | - | 4 | - | 1 | - | - | - | 1 | 5 |
| Temps $M_3$ | - | 4 | 2 | - | 3 | 3 | 4 | - | - |

### 2.8.3 Problème mono-objectif : l'Algorithme Génétique (AG)

Les EAs sont définis par (Merkle, 1997) et (Fogel, 1997). Ces algorithmes sont inspirés de l'évolution des espèces au fil des générations pour s'adapter aux modifications de leur environnement. Cette évolution est basée sur deux opérations : le croisement et la mutation. Le croisement permet l'échange d'informations entre individus pour en créer de nouveaux. La mutation consiste en une modification des individus nouvellement créés par le croisement. Le croisement permet de converger vers la ou les solutions optimales. La mutation évite à l'algorithme de rester dans des optimums locaux. Les AGs sont les formes les plus pures des EAs, sans archives ni hybridations.

Pour expérimenter la MCE, un GA conçu pour le FJSP (Pezzella et al., 2008) sert de base pour l'EA implémenté. La représentation du chromosome (Tay and Wibowo, 2004), la génération de la population initiale (Kacem et al., 2002a) et les opérateurs de croisement (Kacem et al., 2002a) et de mutation (Kyung-Mi et al., 1998) sont spécifiquement adaptés pour le FJSP.

L'algorithme du GA reprend les grandes lignes d'un EA (voir algorithme 2.3). Les paramètres en entrée sont la taille de la population $N$, le nombre de couples $M$, les taux de croisement et de mutation $tc$ et $tm$ et le critère d'arrêt $T$. Le taux de croisement $tc$ détermine la qualité de la convergence de l'algorithme et le taux de mutation $tm$ la capacité de l'algorithme à éviter les minimums locaux. Le critère d'arrêt peut être le nombre de générations ou le temps. Le chromosome est composé de deux parties (Tay and Wibowo, 2004), la première pour l'ordre des opérations sur les machines et la seconde pour l'affectation des opérations aux machines.

À chaque itération, la sélection par rang linéaire est effectuée pour choisir les parents pour la reproduction (Pezzella et al., 2008). Le croisement et la mutation sont faits séparément sur les deux parties du chromosome. Les enfants sont ensuite évalués avec la fonction objectif et triés avec la

**Algorithme 2.3** Algorithme du GA pour le FJSP

**Paramètre** $N$ : Taille de la population
**Paramètre** $M$ : Nombre de couples
**Paramètre** $tc$ : Taux de croisement
**Paramètre** $tm$ : Taux de mutation
**Paramètre** $T$ : Critère d'arrêt

1: $P_0 \leftarrow$ InitialisationPopulation($N$)
2: Evaluation($P_0$)
3: $t \leftarrow 0$
4: **Tant que** $T$ n'est pas atteint **Faire**
5: $C_t \leftarrow$ SélectionRangLinéaire($M$, $P_t$)
6: $E_t \leftarrow$ Croisement($t_c$, $C_t$)
7: $E_t \leftarrow$ Mutation($t_m$, $E_t$)
8: Évaluation($P_t$, $E_t$)
9: $P_{t+1} \leftarrow$ SélectionMeilleurs($N$, $P_t$, $E_t$)
10: $t \leftarrow t + 1$
11: **Fin Tant que**
12: **Retourne** Meilleure solution de $P_t$

population courante en fonction de leurs notes. Les $N$ meilleurs individus sont ensuite sélectionnés pour composer la population à l'itération suivante.

### 2.8.4 Représentation du chromosome

L'objectif du FJSP est l'affectation aux machines et l'ordonnancement des opérations, la représentation du chromosome d'une solution contient les informations dans deux parties distinctes. Cette structure présentée par (Tay and Wibowo, 2004) est utilisée pour coder le chromosome. Les auteurs comparent quatre différentes structures de chromosomes. La plus appropriée est sélectionnée.

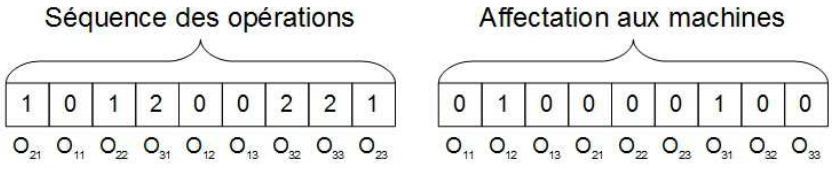


FIGURE 2.7 – Un exemple de représentation du chromosome

La structure est composée de deux vecteurs d'entiers. La Figure 2.7 présente la structure d'un chromosome pour une instance de 3 lots, 9 opérations et 3 machines. La taille de chaque partie est

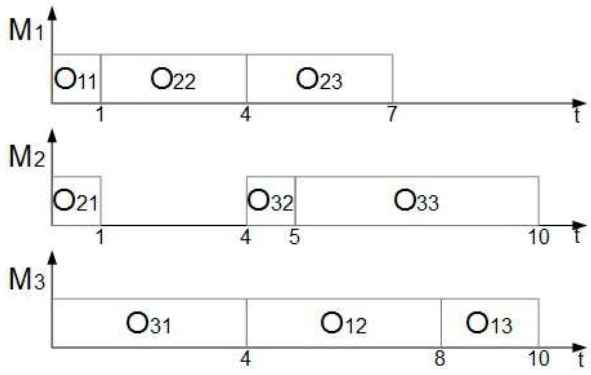


FIGURE 2.8 – Le diagramme de Gantt correspondant à la Figure 2.7

égale au nombre total d'opérations dans l'instance. La première partie, "séquence des opérations", contient l'ordre des opérations de la solution en respectant les contraintes des gammes opératoires de chaque lot. Les opérations des lots sont classées par ordre croissant en fonction du numéro de leur lot et dans l'ordre de la gamme opératoire de ce dernier. La deuxième partie, "affectation aux machines", contient l'affectation des opérations à une machine. Les opérations sont triées par ordre croissant du numéro de leur lot et de leur place dans la gamme opératoire. Pour assurer la faisabilité des solutions, le numéro des machines est modifié pour qu'il soit toujours valide. Cette partie identifie les machines en fonction de leur qualification. Donc, une machine ne sera pas indexée dans l'ensemble des machines capables d'effectuer cette opération, si elle n'est pas qualifiée. Remarque, le vecteur ne contient pas le numéro des machines mais leur identifiant dans l'ensemble des machines qualifiées pour l'opération.

Par exemple, la Figure 2.8 montre le diagramme de Gantt de la solution du chromosome de la Figure 2.7 et de l'instance de la Figure 2.1.

### 2.8.5 Population initiale

La population initiale est générée par des heuristiques afin d'obtenir des solutions de bonne qualité et suffisament variées. Deux heuristiques sont utilisées pour l'affectation des opérations aux machines (Kacem et al., 2002a). Trois autres génèrent le sequançage des opérations sur les machines (Bagheri et al., 2010). La population initiale est générée par des heuristiques, dans les proportions suivantes :

**Étape 1 :** Affectations des opérations :

**Étape 1.1 :** Génération de $pa1 \times N$ affectations initiales par la règle 1.

**Étape 1.2 :** Génération de $pa2 \times N$ affectations initiales par la règle 2.

**Étape 2 :** Séquençage des opérations :

**Étape 2.1 :** Séquencement par la règle RANDOM de $pr \times N$ affectations initiales.

**Étape 2.2 :** Séquencement par la règle MWR de $pmw \times N$ affectations initiales.

**Étape 2.3 :** Séquencement par la régle MOR de $pmo \times N$ affectations initiales.

Où $N$ est de la taille de la population, $pa1$ et $pa2$ sont les pourcentages de solutions construites par les règles d'affectation 1 et 2, $pr$, $pmw$, et $pmo$ sont les pourcentages respectifs des règles de séquencement aléatoire, *Most Work Remaining* (MWR) et *Most Operation Rule* (MOR).

#### 2.8.5.1 Heuristiques d'affectation

Les heuristiques suivantes sont utilisées pour compléter la partie "affectation" du chromosome. La règle 1, la méthode du minimum global, construit itérativement les affectations des opérations aux machines, tandis que la règle 2 introduit une proportion d'aléatoire.

La règle 1 est basée sur l'algorithme 2.4. Cette règle affecte les opérations à une machine tout en tenant compte du temps de réalisation et de la somme du temps de travail des opérations déjà affectées aux machines. L'algorithme 2.4 prend en entrée la matrice $D$ contenant tous les temps de réalisation des opérations sur les machines et la matrice $S$, de même dimension que $D$, initialisée à 0. Il retourne la matrice $S$ où chaque case est égale à 1 ($S_{i,j,k} = 1$) correspond à l'affectation d'une opération à une machine.

Pour éviter que les affectations soit similaires, la règle 2 ou méthode d'affectation par permutations aléatoires présentée par l'algorithme 2.5 est similaire à la règle 1, mais la variable 'Position' est initialisée aléatoirement par une variable aléatoire entre 1 et $m$. Dans les matrices $D$ et $S$, l'ordre des lots est modifié en échangeant la place de deux lots sélectionnés aléatoirement pour donner $D^{'}$. Cet algorithme permet de créer des solutions d'affectation aux machines très différentes les unes des autres.

#### 2.8.5.2 Heuristiques de séquençage

Après l'affection des opérations aux machines, des heuristiques utilisées pour le *Job-Shop Problem* sont adaptées pour établir l'ordre de passage des opérations. Ces trois règles complètent la partie séquence des opérations du premier au dernier bit.

La règle *RANDOM* sélectionne la prochaine opération à effectuer de façon strictement aléatoire tout en respectant le nombre d'opérations de la gamme opératoire de chaque lot.

Pour la règle *Most Work Remaining* (MWR), le lot de la prochaine opération choisie est celui dont la somme du temps de réalisation des opérations encore non ordonnancées est la plus grande.

Dans la règle *Most Operation Remaining* (MOR), le lot de la prochaine opération choisie est celui dont le nombre d'opérations restantes est le plus grand.

**Algorithme 2.4** Règle d'affectation 1 : méthode du minimum global

**Paramètre** $D$

**Paramètre** $S = 0$

**Pour** $i = 1 \rightarrow n$ **Faire**
  **Pour** $j = 1 \rightarrow h_i$ **Faire**
    Min $= +\infty$
    Position $= 1$
    **Pour** $k = 1 \rightarrow m$ **Faire**
      **Si** $d'_{j,i,k} < Min$ **Alors**
        $Min = d'_{j,i,k}$
        $Position = k$
      **Fin Si**
    **Fin Pour**
    $S_{j,i,Position} = 1$
    **Pour** $j' = j + 1 \rightarrow h_i$ **Faire**
      $d'_{j',i,Position} = d'_{j',i,Position} + d_{j,i,Position}$
    **Fin Pour**
    **Pour** $i' = i + 1 \rightarrow n$ **Faire**
      **Pour** $j' = 1 \rightarrow h_{i'}$ **Faire**
        $d'_{j',i',Position} = d'_{j',i',Position} + d_{j,i,Position}$
      **Fin Pour**
    **Fin Pour**
  **Fin Pour**
**Fin Pour**
**Retourne** $S$

### 2.8.6 Opérateurs de sélection

Trois opérateurs de sélection sont abordés dans cette section.

Dans le premier opérateur, la sélection aléatoire choisie les parents au hasard dans la population selon une loi de probabilité uniforme, tous les individus ont la même chance d'être sélectionnés.

Pour le deuxième opérateur, la sélection par tournoi binaire, deux individus sont tirés au hasard, celui qui a la meilleure évaluation est sélectionné comme parent. En cas d'égalité, le nouveau parent est choisi aléatoirement parmi les deux individus.

Avec le dernier opérateur, la sélection par rang linéaire est utilisée pour sélectionner les parents pour la reproduction : croisement et mutation. Les individus sont triés en fonction de leur note obtenue à la fonction objectif. Un rang $r_i \in \{1, ..., N\}$ est affecté à chaque individu, où $N$ est la taille de la population. Par exemple, le rang 1 est affecté au meilleur individu et le rang $N$ pour le

**Algorithme 2.5** Règle d'affectation 2 : méthode d'affectation par permutations aléatoire

**Paramètre** $D^{'}$

**Paramètre** $S = 0$

**Pour** $i = 1 \rightarrow n$ **Faire**
  **Pour** $j = 1 \rightarrow h_i$ **Faire**
    Min = $+\infty$
    r = Random(m)
    Position = r
    **Pour** $k = r \rightarrow m$ **Faire**
      **Si** $d^{'}_{j,i,k} < Min$ **Alors**
        $Min = d^{'}_{j,i,k}$
        $Position = k$
      **Fin Si**
    **Fin Pour**
    **Pour** $k = 1 \rightarrow r - 1$ **Faire**
      **Si** $d^{'}_{j,i,k} < Min$ **Alors**
        $Min = d^{'}_{j,i,k}$
        $Position = k$
      **Fin Si**
    **Fin Pour**
    $S_{j,i,Position} = 1$
    **Pour** $j^{'} = j + 1 \rightarrow h_i$ **Faire**
      $d^{'}_{j^{'},i,Position} = d^{'}_{j^{'},i,Position} + d_{j,i,Position}$
    **Fin Pour**
    **Pour** $i^{'} = i + 1 \rightarrow n$ **Faire**
      **Pour** $j^{'} = 1 \rightarrow h_{i^{'}}$ **Faire**
        $d^{'}_{j^{'},i^{'},Position} = d^{'}_{j^{'},i^{'},Position} + d_{j,i,Position}$
      **Fin Pour**
    **Fin Pour**
  **Fin Pour**
**Fin Pour**
**Retourne** $S$

pire. La probabilité $p_i$ d'être sélectionné pour chaque individu $i$ est déterminée par la formule :

$$p_i = \frac{2.r_i}{N.(N+1)} \quad i = 1, ..., N$$

Un individu avec un rang élevé a plus de chance d'être sélectionné comme parent. Ce système de sélection permet de privilégier les meilleurs individus tout en laissant les plus mauvais une chance de transmettre leur code génétique.

### 2.8.7 Opérateurs de croisement

Les opérateurs de croisement et de mutation définissent l'évolution de la population du GA. Il est préférable pour un opérateur de produire une nouvelle solution réalisable, pour éviter les coûts supplémentaires en calculs générés par la réparation. Dans les problèmes d'ordonnancement, le chromosome représente une solution réalisable et toutes les contraintes entre les opérations sont respectées. Les opérateurs de croisement échangent des informations entre les individus, ce qui permet de trouver de meilleures solutions et à l'algorithme de converger vers la solution optimale. Dans le GA implémenté, deux opérateurs de croisement sont utilisés pour construire de nouvelles solutions réalisables, l'un pour la partie "séquence des opérations" et l'autre pour celle "affectation aux machines". Ces deux opérateurs respectent les contraintes sur les opérations pour fournir de nouvelles solutions faisables. Dans le GA, le croisement se fait séparément sur les deux parties du chromosome. Pour chaque partie, un nombre aléatoire $p \in [0, 1]$ est généré aléatoirement pour chaque partie, si $p < tc$ alors le croisement est effectué, sinon la partie du parent est recopiée dans celle de l'enfant.

Pour la partie "séquence des opérations", le *precedence-preserving order-based crossover* (POX) (Kacem et al., 2002a) est utilisé. Le POX génère deux nouveaux chromosomes $C_1$ et $C_2$ à partir de deux autres existant $P_1$ et $P_2$ selon la procédure suivante :

**Étape 1 :** Sélectionner aléatoirement un gène qui correspond à une opération $O_{ij}$ dans la partie "séquence des opérations" du parent $P_1$.

**Étape 2 :** Trouver les places des opérations du lot $J_i$ sur $P_1$.

**Étape 3 :** Copier les opérations du lot dans l'enfant $C_1$ aux mêmes places que dans le parent $P_1$.

**Étape 4 :** Copier les opérations restantes dans l'enfant $C_1$ dans le même ordre qu'il apparaissent dans le parent $P_2$.

Pour créer l'enfant $C_2$ on utilise la même procédure, mais en sélectionnant une opération dans le parent $P_2$ et on complète en prenant les opérations du parent $P_1$. Avec ce procédé, les contraintes sur l'ordre des opérations sont respectées et ainsi les nouvelles solutions sont réalisables.

L'exemple suivant montre le fonctionnement du POX pour la création d'un enfant. Les opérations du lot 1 sont sélectionnées aléatoirement.

$$P_1 : (\ \boxed{1}\ 3\ \boxed{1}\ 2\ 2\ 3\ \boxed{1}\ 2\ 3\ )$$

$$P_2 : (\ 3\ 1\ 3\ 2\ 1\ 2\ 3\ 1\ 2\ )$$

$$C_1 : (\ \boxed{1}\ 3\ \boxed{1}\ 3\ 2\ 2\ \boxed{1}\ 3\ 2\ )$$

Pour la partie "affectation aux machines", un classique croisement à un point est effectué. La procédure du croisement à un point est détaillée ci-dessous :

**Étape 1 :** Générer un nombre aléatoire $r$, $2 \leq r \leq (n-1)$, où $n$ est le nombre d'opérations dans l'instance.

**Étape 2 :** Sélectionner tous les gènes entre 1 et $r$ du parent $P_1$ et les copier à la même place dans l'enfant $C_1$

**Étape 3 :** Compléter l'enfant $C_1$ en prenant les gènes entre $r+1$ et $n$ du parent $P_2$ et en les recopiant à la même place.

La même procédure est utilisée pour créer l'enfant $C_2$ mais en inversant les rôles des parents $P_1$ et $P_2$.

L'exemple suivant montre le fonctionnement du croisement à un point pour la création d'un enfant. La position $r$ est sélectionnée aléatoirement.

$$P_1 : (\ \boxed{1\ 2\ 1\ 1\ 2}\ 1\ 2\ 1\ 1\ )$$

$$P_2 : (\ 1\ 2\ 1\ 1\ 1\ \boxed{2\ 2\ 1\ 1}\ )$$

$$C_1 : (\ 1\ 2\ 1\ 1\ 2\ 2\ 2\ 1\ 1\ )$$

### 2.8.8 Opérateurs de mutation

Les opérateurs de mutation modifient légèrement une solution. La mutation permet d'introduire de nouvelles caractéristiques dans la population augmentant ainsi la diversité des individus. La diversité de la population permet au GA d'éviter les minimums locaux. Les deux opérateurs de mutation utilisés pour modifier une solution s'appliquent pour l'un sur la partie "séquence des opérations" et pour l'autre sur la partie "affectation aux machines". Ces opérateurs respectent aussi les contraintes du problème pour les solutions modifiées restent faisables. La mutation est opérée sur les deux parties du chromosome.

Pour la partie "séquence des opérations", le *precedence-preserving shift mutation* (PPS) (Kacem et al., 2002a) est utilisé. Soit $X = \{x_1, x_2, ..., x_n\}$, avec $n$ le nombre de gènes/opérations, la partie "séquence des opérations" d'un chromosome qui candidate pour muter. La procédure suivante est effectuée pour chaque gène du chromosome $X$ :

**Étape 1 :** Pour le géne $x_i$ à la position $i$, générer un nombre aléatoire $r \in [0, 1]$.

**Étape 2 :** Si $r > tm$, arrêter la procédure et passer au gène suivant, sinon continuer.

**Étape 3 :** Trouver les places des opérations précédente $lmp$ et suivante $rmp$ sur la gamme opératoire du lot par rapport à celle de l'opération du gène $x_i$ .

$$\begin{aligned} lmp &= min_k\{k|\{x_j|j = k+1, ..., i-1\} \neq x_i\} \\ rmp &= max_k\{k|\{x_j|j = i+1, ..., n-1\} \neq x_i\} \end{aligned}$$

avec $k \in \{1, ..., n\}$.

**Étape 4 :** Sélectionner aléatoirement une position $p \in [lmp, rmp]$.

**Étape 5 :** Déplacer le gène $x_i$ à la position $p$.

L'exemple suivant montre comment le PPS fonctionne. La position 5 est choisie aléatoirement, elle correspond au lot $x_i = 3$. $lmp$ est à la deuxième position et $rmp$ à la huitième position. La nouvelle de $x_i$ est $p = 7$. Le chromosome est modifié comme suit :

$$P : (\ 2\ \underline{3}\ 1\ 1\ \boxed{3}\ 2\ 2\ \underline{3}\ 1\ )$$
$$C : (\ 2\ \underline{3}\ 1\ 1\ 2\ 2\ \boxed{3}\ \underline{3}\ 1\ )$$

Pour la partie "affectation aux machines", un changement classique est effectué sur l'un des gènes. Cette mutation consiste à changer d'opération de machine. La procédure détaillée ci-dessous est effectuée pour chaque gène :

**Étape 1 :** Pour l'opération $O_{ij}$ à la position $p$, générer un nombre aléatoire $r \in [0, 1]$.

**Étape 2 :** Si $r > tm$, arrêter la procédure et passer au gène suivant, sinon continuer

**Étape 3 :** Choisir un nombre aléatoire $m_2$, $m_2 \in \{1, |M_{ij}|\}/\{m_1\}$, $m_1$ est la valeur actuelle du gène $x_p$ et $|M_{ij}|$ est le nombre de machine dans l'ensemble $M_{ij}$. $M_{ij}$ est l'ensemble des machines qualifiées pour faire $O_{ij}$.

**Étape 4 :** Remplacer $m_1$ par $m_2$.

L'exemple suivant montre le fonctionnement de la mutation par changement d'affectation. Le gène à la position 6 est choisi pour muter. Après un tirage aléatoire, l'opération du gène passe de la machine 1 à la machine 2. Le chromosome est modifié comme suit :

$$P : (\ 1\ 2\ 1\ 1\ 2\ \boxed{1}\ 2\ 1\ 1\ )$$
$$C : (\ 1\ 2\ 1\ 1\ 2\ \boxed{2}\ 2\ 1\ 1\ )$$

### 2.8.9 Résultats expérimentaux

Le GA (Pezzella et al., 2008) décrit dans la section 2.8.3 est implémenté sur un processeur core i5 3,4 GHz et 8 Giga de RAM, avec le système d'exploitation Windows 7. Les instances tirées de la littérature utilisées sont présentées, suivis par les paramètres et les hybridations utilisées et enfin les résultats commentés. Le GA et l'hybridation par la MCE sont testés sur trois ensembles contenant 15, 18 et 40 instances.

#### 2.8.9.1 Instances

Des générateurs d'instances et des paramètres de la littérature sont employés (Brandimarte, 1993), (Dauzère-Pérès and Paulli, 1997) et (Adams et al., 1988).

L'ensemble des données de (Brandimarte, 1993) contient 15 problèmes avec un nombre de lots $n$ entre 10 et 30, un nombre de machines $m$ entre 4 et 15 et un nombre d'opérations $h$ par lot compris entre 5 et 15. Le temps de réalisation $p$ d'une opération est compris entre 2 et 6. Entre 2 et 6 machines sont qualifiées pour réaliser une opération. Les instances sont générées aléatoirement selon une loi à distribution uniforme en respectant les paramètres du tableau 2.2.

TABLEAU 2.2 – Les instances de (Brandimarte, 1993)

| | n | m | h | p | q |
|---|---|---|---|---|---|
| mk1 | 10 | 6 | 5-7 | 3 | 1-7 |
| mk2 | 10 | 6 | 5-7 | 6 | 1-7 |
| mk3 | 15 | 8 | 10-10 | 5 | 1-20 |
| mk4 | 15 | 8 | 3-10 | 3 | 1-10 |
| mk5 | 15 | 4 | 5-10 | 2 | 5-10 |
| mk6 | 10 | 15 | 15-15 | 5 | 1-10 |
| mk7 | 20 | 5 | 5-5 | 5 | 1-20 |
| mk8 | 20 | 10 | 10-15 | 2 | 5-20 |
| mk9 | 20 | 10 | 10-15 | 5 | 5-20 |
| mk10 | 20 | 15 | 10-15 | 5 | 5-20 |
| mk11 | 30 | 5 | 5-8 | 2 | 10-30 |
| mk12 | 30 | 10 | 5-10 | 2 | 10-30 |
| mk13 | 30 | 10 | 5-10 | 5 | 10-30 |
| mk14 | 30 | 15 | 8-12 | 2 | 10-30 |
| mk15 | 30 | 15 | 8-12 | 5 | 10-30 |

Dauzère-Pérès a proposé un ensemble de 18 instances (Dauzère-Pérès and Paulli, 1997) avec un nombre de lots $n$ entre 10 et 20, un nombre de machines $m$ entre 5 et 10 et un nombre d'opérations $h$ entre 15 et 25 sur chaque lot. Le temps de réalisation d'une opération est compris entre 10 et 100 avec une variation de 5 selon la machine qualifiée. Le nombre de machines qualifiées pour faire une opération est au minimum de 1. Lors de la construction de l'instance, une machine est qualifiée pour une opération selon une probabilité $p$. Une loi de distribution uniforme est utilisée pour générer les instances en respectant les paramètres du tableau 2.3.

TABLEAU 2.3 – Les instances de (Dauzère-Pérès and Paulli, 1997)

| | n | m | h | p | $\Delta p_i$ | P |
|---|---|---|---|---|---|---|
| 1a | 10 | 5 | 15-25 | 10-100 | 0 | 0.1 |
| 2a | 10 | 5 | 15-25 | 10-100 | 0 | 0.3 |
| 3a | 10 | 5 | 15-25 | 10-100 | 0 | 0.5 |
| 4a | 10 | 5 | 15-25 | 10-100 | 5 | 0.1 |
| 5a | 10 | 5 | 15-25 | 10-100 | 5 | 0.3 |
| 6a | 10 | 5 | 15-25 | 10-100 | 5 | 0.5 |
| 7a | 15 | 8 | 15-25 | 10-100 | 0 | 0.1 |
| 8a | 15 | 8 | 15-25 | 10-100 | 0 | 0.3 |
| 9a | 15 | 8 | 15-25 | 10-100 | 0 | 0.5 |
| 10a | 15 | 8 | 15-25 | 10-100 | 5 | 0.1 |
| 11a | 15 | 8 | 15-25 | 10-100 | 5 | 0.3 |
| 12a | 15 | 8 | 15-25 | 10-100 | 5 | 0.5 |
| 13a | 20 | 10 | 20-25 | 10-100 | 0 | 0.1 |
| 14a | 20 | 10 | 20-25 | 10-100 | 0 | 0.3 |
| 15a | 20 | 10 | 20-25 | 10-100 | 0 | 0.5 |
| 16a | 20 | 10 | 20-25 | 10-100 | 5 | 0.1 |
| 17a | 20 | 10 | 20-25 | 10-100 | 5 | 0.3 |
| 18a | 20 | 10 | 20-25 | 10-100 | 5 | 0.5 |

Adams a adapté des instances du *Job-Shop Problem* (JSP) (Adams et al., 1988) et (Fisher and Thompson, 1963) au FJSP. Quarante instances sont sélectionnées avec un nombre de lots $n$ entre 10 et 30, un nombre de machines $m$ entre 5 et 15 et le nombre d'opérations $h$ sur chaque lot est égal à celui du nombre de machines dans l'instance. Le temps de réalisation $p$ des opérations est entre 5 et 99. Deux règles (Hurink et al., 1994) sont utilisées pour calculer le nombre de machines qualifiées $q$ par opérations (voir tableau 2.4). Les instances sont générées aléatoirement en suivant une loi de distribution uniforme avec comme paramètres les valeurs du tableau 2.5. Les paramètres des lois

TABLEAU 2.4 – Exemple de population avec son évaluation

| | $\lvert M_{ij}\rvert_{min}$ | $\lvert M_{ij}\rvert_{moy}$ | $\lvert M_{ij}\rvert_{max}$ |
|---|---|---|---|
| rdata | 1 | 2 | 3 |
| vdata | $max(1, \frac{1}{5}m)$ | $\frac{1}{2}m$ | $\frac{4}{5}m$ |

utilisées pour transformer une instance de JSP en FJSP sont dans le tableau 2.4, les noms des lois sont dans la première colonne. Les autres colonnes contiennent dans l'ordre le nombre minimum, moyen et maximum de machines qualifiées.

TABLEAU 2.5: Les instances d'(Adams et al., 1988)

| Les instances d'Adams | | | | | | |
|---|---|---|---|---|---|---|
| | n | m | h | p | q (rdata \| vdata) | |
| 101 | 10 | 5 | 5-5 | 5-99 | 1-3 | 1-4 |
| 102 | 10 | 5 | 5-5 | 5-99 | 1-3 | 1-4 |
| 103 | 10 | 5 | 5-5 | 5-99 | 1-3 | 1-4 |
| 104 | 10 | 5 | 5-5 | 5-99 | 1-3 | 1-4 |
| 105 | 10 | 5 | 5-5 | 5-99 | 1-3 | 1-4 |
| 106 | 15 | 5 | 5-5 | 5-99 | 1-3 | 1-4 |
| 107 | 15 | 5 | 5-5 | 5-99 | 1-3 | 1-4 |
| 108 | 15 | 5 | 5-5 | 5-99 | 1-3 | 1-4 |
| 109 | 15 | 5 | 5-5 | 5-99 | 1-3 | 1-4 |
| 110 | 15 | 5 | 5-5 | 5-99 | 1-3 | 1-4 |
| 111 | 20 | 5 | 5-5 | 5-99 | 1-3 | 1-4 |
| 112 | 20 | 5 | 5-5 | 5-99 | 1-3 | 1-4 |
| 113 | 20 | 5 | 5-5 | 5-99 | 1-3 | 1-4 |
| 114 | 20 | 5 | 5-5 | 5-99 | 1-3 | 1-4 |
| 115 | 20 | 5 | 5-5 | 5-99 | 1-3 | 1-4 |
| 116 | 10 | 10 | 10-10 | 5-99 | 1-3 | 2-8 |
| 117 | 10 | 10 | 10-10 | 5-99 | 1-3 | 2-8 |
| 118 | 10 | 10 | 10-10 | 5-99 | 1-3 | 2-8 |
| 119 | 10 | 10 | 10-10 | 5-99 | 1-3 | 2-8 |

| Les instances d'Adams | | | | | | |
|---|---|---|---|---|---|---|
| | n | m | h | p | q (rdata \| vdata) | |
| 120 | 10 | 10 | 10-10 | 5-99 | 1-3 | 2-8 |
| 121 | 15 | 10 | 10-10 | 5-99 | 1-3 | 2-8 |
| 122 | 15 | 10 | 10-10 | 5-99 | 1-3 | 2-8 |
| 123 | 15 | 10 | 10-10 | 5-99 | 1-3 | 2-8 |
| 124 | 15 | 10 | 10-10 | 5-99 | 1-3 | 2-8 |
| 125 | 15 | 10 | 10-10 | 5-99 | 1-3 | 2-8 |
| 126 | 20 | 10 | 10-10 | 5-99 | 1-3 | 2-8 |
| 127 | 20 | 10 | 10-10 | 5-99 | 1-3 | 2-8 |
| 128 | 20 | 10 | 10-10 | 5-99 | 1-3 | 2-8 |
| 129 | 20 | 10 | 10-10 | 5-99 | 1-3 | 2-8 |
| 130 | 20 | 10 | 10-10 | 5-99 | 1-3 | 2-8 |
| 131 | 30 | 10 | 10-10 | 5-99 | 1-3 | 2-8 |
| 132 | 30 | 10 | 10-10 | 5-99 | 1-3 | 2-8 |
| 133 | 30 | 10 | 10-10 | 5-99 | 1-3 | 2-8 |
| 134 | 30 | 10 | 10-10 | 5-99 | 1-3 | 2-8 |
| 135 | 30 | 10 | 10-10 | 5-99 | 1-3 | 2-8 |
| 136 | 15 | 15 | 15-15 | 5-99 | 1-3 | 3-12 |
| 137 | 15 | 15 | 15-15 | 5-99 | 1-3 | 3-12 |
| 138 | 15 | 15 | 15-15 | 5-99 | 1-3 | 3-12 |
| 139 | 15 | 15 | 15-15 | 5-99 | 1-3 | 3-12 |
| 140 | 15 | 15 | 15-15 | 5-99 | 1-3 | 3-12 |

Dans les tableaux 2.2, 2.3 et 2.5, chaque ligne contient les paramètres pour une instance. La première colonne contient les noms des instances, la colonne $n$ le nombre de lots, la colonne $m$ le nombre de machines, la colonne $h$ le nombre minimal et maximal d'opérations de la gamme opératoire d'un lot, la colonne $p$ les valeurs minimum et maximum du temps de réalisation d'une opération et la colonne $q$ les nombres minimum et maximum de machines qualifiées par opérations. Pour le tableau 2.3, la colonne $\Delta p_i$ détermine la différence de temps de réalisation pour une opération sur les machines qualifiées et la colonne $P$ le taux de machines qualifiées pour faire une opération.

#### 2.8.9.2 Paramétrages

Les individus de la population initiale sont créés par plusieurs heuristiques ; chacune est utilisée pour un pourcentage défini comme suit :

— taux d'affectation selon la méthode du minimum global : 10%
— taux d'affectation selon la méthode de permutation aléatoire : 90%
— taux de séquençages aléatoires : 20%
— taux de séquençages selon la loi MWR : 40%
— taux de séquençages selon la loi MOR : 40%

Les paramètres retenus pour le GA sont les suivants :

— taille de la population : 100
— nombre de générations : 2000
— probabilité pour le croisement POX : 45%
— probabilité pour le croisement d'affectation : 45%
— probabilité pour la mutation PPS : 2%
— probabilité pour la mutation d'affectation : 2%

La sélection par rang linéaire est retenue pour choisir les parents à chaque itération.

Les hybridations expérimentées sont les GAs avec la recherche locale sur la meilleure solution (GA-LS) et sur les solutions dans les zones *nd* (GA-Mce1). La dernière hybridation pratique une combinaison de création de solutions dans les zones *zi* et l'application d'une recherche locale sur ces nouvelles solutions et sur celles dans les zones *nd* (GA-Mce4). Pour les hybridations GA-(Mce1,Mce4) un indicateur décide périodiquement de l'application de la PRL. Cet indicateur demande une PRL si la progression de la note de la meilleure solution est faible. Ensuite, la HC est appliquée sur les individus concernés et s'arrête après un nombre d'explorations du voisinage sans trouver de solutions améliorantes.

— période d'application d'une PRL : $T = 100$.
— nombre d'essais infructueux avant arrêt pour la LS : 500.
— indicateur de décision : $Ind - Mce = 10\%$.
— nombre d'essais pour la création de solutions dans les *zi* : 1000.

Le critère d'arrêt des hybridations est le temps mis par l'algorithme initial pour effectuer 2000 générations.

Les résultats dans les tableaux sont la moyenne de 10 simulations. Les tableaux 2.6, 2.7, 2.8 et 2.9 contiennent les résultats de la convergence pour les instances de Barndimarte, Dauzère-Pérès et Adams (voir tableaux 2.2, 2.3 et 2.5). La première colonne contient le nom et la taille des instances, la seconde le nom des algorithmes et la dernière l'évolution de la convergence des algorithmes. Chaque ligne contient les résultats de la convergence de quatre algorithmes (GA, GA-LS, GA-Mce1 et GA-Mce4) pour une instance.

TABLEAU 2.6: Convergence des algorithmes pour une simulation pour les instances de (Brandimarte, 1993)

| Convergence des algorithmes pour une simulation pour les instances de Brandimarte | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | | |
| mk1 | GA | 69 | 53 | 46 | 45 | 45 | 45 | 43 | 43 | 43 | 43 | 42 | 41 | 41 | 41 | 41 | 40 | 40 | 40 | 40 | 40 | |
| (6/7/10) | GA-LS | 37 | 37 | 37 | 37 | 37 | 37 | 37 | 37 | 37 | 37 | 37 | 37 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | | |
| | GA-Mce1 | 37 | 37 | 37 | 37 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | | | |
| | GA-Mce4 | 37 | 37 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | 35 | | | |
| mk2 | GA | 68 | 50 | 45 | 41 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 39 | 38 | 38 | 38 | 38 | 38 | 38 | 38 | |
| (6/7/10) | GA-LS | 32 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 30 | 30 | | |
| | GA-Mce1 | 32 | 32 | 31 | 31 | 31 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | | | |
| | GA-Mce4 | 32 | 32 | 31 | 31 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | | | |
| mk3 | GA | 389 | 263 | 245 | 244 | 242 | 241 | 238 | 234 | 232 | 228 | 224 | 222 | 222 | 221 | 221 | 221 | 218 | 215 | 214 | 213 | |
| (8/10/15) | GA-LS | 188 | 174 | 174 | 168 | 168 | 163 | 163 | 163 | 162 | 162 | 162 | 159 | 159 | 159 | 159 | 159 | 159 | 159 | 159 | | |
| | GA-Mce1 | 190 | 176 | 176 | 170 | 168 | 167 | 165 | 163 | 163 | 163 | 161 | 161 | 160 | 160 | 159 | 158 | 158 | 158 | 157 | | |
| | GA-Mce4 | 190 | 176 | 176 | 173 | 164 | 164 | 164 | 164 | 164 | 164 | 164 | 164 | 162 | 161 | 161 | 161 | 160 | 159 | 156 | | |
| mk4 | GA | 123 | 94 | 90 | 89 | 88 | 88 | 85 | 84 | 84 | 83 | 83 | 83 | 82 | 82 | 82 | 78 | 78 | 78 | 77 | 77 | |
| (8/10/15) | GA-LS | 62 | 61 | 61 | 61 | 60 | 60 | 60 | 60 | 60 | 60 | 60 | 59 | 59 | 59 | 59 | 59 | 59 | 59 | 59 | | |
| | GA-Mce1 | 64 | 63 | 61 | 61 | 61 | 59 | 59 | 59 | 59 | 59 | 59 | 58 | 58 | 58 | 58 | 58 | 58 | 58 | 57 | | |
| | GA-Mce4 | 64 | 63 | 62 | 62 | 62 | 59 | 57 | 57 | 57 | 57 | 57 | 57 | 57 | 57 | 57 | 57 | 57 | 57 | | | |
| mk5 | GA | 289 | 245 | 237 | 237 | 237 | 237 | 231 | 230 | 230 | 226 | 226 | 225 | 225 | 225 | 224 | 224 | 224 | 223 | 223 | 223 | |
| (4/10/15) | GA-LS | 218 | 216 | 212 | 211 | 210 | 208 | 206 | 206 | 206 | 206 | 204 | 204 | 204 | 204 | 204 | 204 | 204 | 204 | 204 | | |
| | GA-Mce1 | 218 | 216 | 212 | 207 | 205 | 205 | 205 | 204 | 204 | 204 | 204 | 204 | 204 | 204 | 204 | 204 | 204 | 204 | | | |
| | GA-Mce4 | 218 | 216 | 216 | 209 | 209 | 209 | 208 | 207 | 206 | 206 | 206 | 206 | 206 | 206 | 202 | 202 | 202 | 202 | | | |
| mk6 | GA | 172 | 135 | 127 | 127 | 126 | 123 | 123 | 121 | 121 | 114 | 114 | 114 | 114 | 114 | 112 | 112 | 112 | 112 | 109 | 109 | |

| Convergence des algorithmes pour une simulation pour les instances de Brandimarte (suite) | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | | |
| (15/5/10) | GA-LS | 128 | 127 | 126 | 125 | 122 | 122 | 118 | 116 | 115 | 114 | 114 | 114 | 110 | 108 | 107 | 107 | 104 | 103 | 103 | | |
| | GA-Mce1 | 135 | 127 | 127 | 126 | 124 | 117 | 116 | 116 | 110 | 108 | 108 | 105 | 104 | 104 | 102 | 97 | 95 | 95 | | | |
| | GA-Mce4 | 135 | 127 | 127 | 125 | 124 | 121 | 120 | 120 | 114 | 114 | 103 | 102 | 97 | 97 | 97 | 97 | 97 | 97 | | | |
| mk7 | GA | 278 | 193 | 186 | 186 | 186 | 186 | 186 | 181 | 181 | 180 | 179 | 178 | 174 | 174 | 174 | 171 | 171 | 171 | 171 | 171 | |
| (5/5/20) | GA-LS | 165 | 155 | 155 | 155 | 155 | 155 | 155 | 155 | 155 | 155 | 155 | 155 | 155 | 155 | 153 | 151 | 151 | 151 | 151 | | |
| | GA-Mce1 | 166 | 161 | 159 | 159 | 159 | 159 | 159 | 158 | 158 | 158 | 158 | 158 | 158 | 157 | 154 | 151 | 151 | 151 | | | |
| | GA-Mce4 | 166 | 161 | 159 | 159 | 152 | 152 | 152 | 152 | 152 | 152 | 152 | 151 | 151 | 150 | 150 | 150 | 150 | 150 | | | |
| mk8 | GA | 659 | 508 | 481 | 466 | 466 | 464 | 463 | 448 | 434 | 432 | 428 | 428 | 428 | 428 | 428 | 428 | 425 | 425 | 421 | 421 | |
| (10/15/20) | GA-LS | 410 | 405 | 405 | 405 | 403 | 400 | 395 | 392 | 389 | 388 | 388 | 388 | 388 | 387 | 387 | 387 | 383 | 383 | 383 | | |
| | GA-Mce1 | 418 | 406 | 401 | 395 | 389 | 383 | 379 | 379 | 378 | 378 | 378 | 378 | 378 | 378 | 378 | 378 | 378 | 378 | 376 | | |
| | GA-Mce4 | 418 | 406 | 401 | 391 | 391 | 388 | 378 | 376 | 376 | 375 | 373 | 373 | 373 | 373 | 373 | 372 | 372 | 372 | | | |
| mk9 | GA | 646 | 488 | 465 | 459 | 454 | 449 | 449 | 448 | 439 | 435 | 434 | 432 | 426 | 424 | 424 | 424 | 416 | 416 | 416 | 416 | |
| (10/15/20) | GA-LS | 390 | 374 | 371 | 364 | 349 | 345 | 344 | 344 | 341 | 341 | 341 | 341 | 341 | 340 | 340 | 340 | 340 | 340 | 340 | | |
| | GA-Mce1 | 390 | 380 | 376 | 361 | 361 | 361 | 346 | 340 | 337 | 335 | 335 | 334 | 334 | 333 | 333 | 333 | 333 | 333 | | | |
| | GA-Mce4 | 390 | 380 | 376 | 361 | 361 | 360 | 349 | 349 | 341 | 340 | 338 | 337 | 336 | 334 | 334 | 334 | 334 | 334 | | | |
| mk10 | GA | 491 | 395 | 373 | 360 | 355 | 353 | 352 | 347 | 343 | 342 | 338 | 335 | 335 | 334 | 334 | 334 | 334 | 329 | 323 | 322 | |
| (15/15/20) | GA-LS | 280 | 266 | 265 | 264 | 256 | 254 | 253 | 253 | 253 | 253 | 253 | 253 | 253 | 253 | 253 | 253 | 253 | 253 | 253 | | |
| | GA-Mce1 | 280 | 270 | 267 | 260 | 256 | 251 | 251 | 250 | 249 | 249 | 247 | 247 | 246 | 246 | 246 | 246 | 246 | 246 | | | |
| | GA-Mce4 | 280 | 270 | 267 | 262 | 258 | 258 | 257 | 254 | 250 | 250 | 248 | 246 | 246 | 246 | 246 | 246 | 246 | 245 | | | |
| mk11 | GA | 1065 | 877 | 860 | 857 | 856 | 853 | 851 | 848 | 847 | 846 | 845 | 845 | 844 | 844 | 844 | 842 | 842 | 834 | 832 | 832 | |
| (5/8/30) | GA-LS | 854 | 824 | 814 | 812 | 800 | 799 | 799 | 799 | 799 | 799 | 799 | 798 | 798 | 798 | 798 | 798 | 798 | 798 | 798 | | |
| | GA-Mce1 | 854 | 824 | 813 | 809 | 801 | 797 | 794 | 792 | 792 | 792 | 792 | 792 | 792 | 792 | 792 | 792 | 792 | 792 | | | |
| | GA-Mce4 | 854 | 824 | 813 | 803 | 801 | 791 | 791 | 791 | 791 | 791 | 791 | 791 | 791 | 791 | 791 | 791 | 791 | 791 | | | |
| mk12 | GA | 846 | 682 | 666 | 646 | 640 | 640 | 640 | 634 | 629 | 629 | 615 | 609 | 606 | 602 | 587 | 582 | 581 | 581 | 581 | 581 | |

| Convergence des algorithmes pour une simulation pour les instances de Brandimarte (suite) | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | | |
| (10/10/30) | GA-LS | 598 | 576 | 576 | 568 | 563 | 563 | 561 | 554 | 554 | 554 | 554 | 554 | 554 | 554 | 554 | 554 | 554 | 554 | 554 | | |
| | GA-Mce1 | 598 | 586 | 576 | 576 | 563 | 563 | 550 | 541 | 541 | 541 | 541 | 541 | 541 | 541 | 541 | 541 | 541 | 541 | 541 | | |
| | GA-Mce4 | 598 | 586 | 576 | 576 | 576 | 576 | 576 | 576 | 576 | 576 | 576 | 576 | 576 | 576 | 563 | 563 | 563 | 551 | 541 | | |
| mk13 | GA | 778 | 632 | 609 | 582 | 566 | 562 | 562 | 561 | 552 | 550 | 533 | 517 | 516 | 516 | 502 | 500 | 500 | 500 | 494 | 492 | |
| (10/10/30) | GA-LS | 450 | 444 | 434 | 422 | 422 | 422 | 414 | 413 | 405 | 403 | 400 | 400 | 400 | 400 | 400 | 400 | 400 | 400 | 398 | | |
| | GA-Mce1 | 458 | 443 | 435 | 419 | 419 | 412 | 408 | 408 | 407 | 402 | 402 | 401 | 400 | 400 | 396 | 396 | 395 | 395 | | | |
| | GA-Mce4 | 458 | 443 | 435 | 419 | 419 | 419 | 419 | 408 | 407 | 407 | 407 | 407 | 407 | 407 | 407 | 407 | 395 | 395 | | | |
| mk14 | GA | 1039 | 816 | 804 | 785 | 772 | 757 | 757 | 755 | 745 | 741 | 737 | 725 | 725 | 709 | 706 | 706 | 703 | 695 | 694 | 694 | |
| (15/12/30) | GA-LS | 712 | 698 | 684 | 684 | 684 | 684 | 684 | 684 | 684 | 684 | 684 | 684 | 684 | 673 | 673 | 673 | 673 | 673 | 673 | | |
| | GA-Mce1 | 712 | 700 | 686 | 684 | 684 | 684 | 684 | 684 | 684 | 679 | 673 | 673 | 673 | 666 | 666 | 666 | 665 | 663 | 662 | | |
| | GA-Mce4 | 712 | 700 | 686 | 684 | 671 | 669 | 665 | 662 | 662 | 662 | 662 | 653 | 653 | 653 | 653 | 653 | 653 | 651 | | | |
| mk15 | GA | 593 | 504 | 487 | 468 | 462 | 456 | 456 | 454 | 451 | 447 | 444 | 441 | 438 | 438 | 438 | 438 | 438 | 437 | 437 | 436 | |
| (15/12/30) | GA-LS | 518 | 504 | 489 | 478 | 467 | 461 | 461 | 461 | 458 | 457 | 457 | 453 | 453 | 449 | 448 | 444 | 441 | 439 | 438 | | |
| | GA-Mce1 | 520 | 495 | 483 | 477 | 461 | 460 | 459 | 457 | 452 | 447 | 444 | 436 | 435 | 434 | 432 | 431 | 431 | 431 | 429 | | |
| | GA-Mce4 | 520 | 495 | 483 | 477 | 461 | 460 | 460 | 460 | 460 | 458 | 458 | 456 | 453 | 447 | 445 | 445 | 443 | 442 | | | |

TABLEAU 2.7: Convergence des algorithmes pour une simulation pour les instances de (Dauzère-Pérès and Paulli, 1997)

| Convergence des algorithmes pour une simulation pour les instances de Dauzère-Pérès | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | | |
| 1a | GA | 4408 | 3499 | 3472 | 3445 | 3445 | 3445 | 3414 | 3398 | 3395 | 3392 | 3392 | 3392 | 3392 | 3392 | 3392 | 3392 | 3392 | 3389 | 3389 | 3377 | |
| (5/25/10) | GA-LS | 3499 | 3412 | 3393 | 3393 | 3385 | 3385 | 3385 | 3385 | 3385 | 3385 | 3385 | 3376 | 3374 | 3374 | 3374 | 3374 | 3374 | | | | |
| | GA-Mce1 | 499 | 3412 | 3411 | 3377 | 3374 | 3374 | 3374 | 3354 | 3341 | 3341 | 3341 | 3339 | 3330 | 3330 | | | | | | | |

| Convergence des algorithmes pour une simulation pour les instances de Dauzère-Pérès (suite) | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | |
| | GA-Mce4 | 3499 | 3412 | 3387 | 3387 | 3362 | 3362 | 3362 | 3362 | 3362 | 3362 | 3362 | | | | | | | | | |
| 2a | GA | 3915 | 3013 | 2886 | 2828 | 2812 | 2805 | 2782 | 2782 | 2782 | 2782 | 2782 | 2782 | 2782 | 2782 | 2782 | 2762 | 2755 | 2749 | 2746 | 2742 |
| (5/25/10) | GA-LS | 3013 | 2886 | 2850 | 2812 | 2789 | 2780 | 2722 | 2722 | 2722 | 2722 | 2722 | 2722 | 2722 | 2722 | 2722 | 2722 | 2722 | 2722 | 2722 | |
| | GA-Mce1 | 3013 | 2886 | 2850 | 2812 | 2789 | 2755 | 2729 | 2729 | 2684 | 2680 | 2680 | 2680 | 2680 | 2678 | 2668 | 2666 | 2658 | 2658 | | |
| | GA-Mce4 | 3013 | 2886 | 2850 | 2812 | 2789 | 2773 | 2730 | 2730 | 2706 | 2681 | 2663 | 2663 | 2663 | 2663 | 2663 | 2663 | | | | |
| 3a | GA | 3488 | 2643 | 2567 | 2511 | 2501 | 2491 | 2472 | 2470 | 2463 | 2460 | 2460 | 2460 | 2431 | 2431 | 2431 | 2431 | 2422 | 2417 | 2411 | 2403 |
| (5/25/10) | GA-LS | 2643 | 2567 | 2511 | 2501 | 2491 | 2485 | 2479 | 2449 | 2449 | 2446 | 2439 | 2436 | 2433 | 2426 | 2404 | 2393 | 2393 | 2393 | | |
| | GA-Mce1 | 2643 | 2567 | 2511 | 2501 | 2477 | 2471 | 2436 | 2431 | 2428 | 2428 | 2411 | 2394 | 2358 | 2333 | 2330 | 2326 | | | | |
| | GA-Mce4 | 2643 | 2567 | 2511 | 2501 | 2472 | 2436 | 2432 | 2429 | 2406 | 2369 | 2369 | 2346 | 2346 | 2344 | 2344 | 2344 | | | | |
| 4a | GA | 4302 | 3546 | 3461 | 3461 | 3438 | 3438 | 3438 | 3438 | 3438 | 3415 | 3414 | 3414 | 3414 | 3414 | 3412 | 3402 | 3402 | 3401 | 3401 | 3401 |
| (5/25/10) | GA-LS | 3546 | 3461 | 3461 | 3438 | 3438 | 3438 | 3414 | 3407 | 3407 | 3407 | 3407 | 3407 | 3405 | 3393 | 3347 | 3345 | 3345 | 3345 | 3343 | |
| | GA-Mce1 | 3546 | 3461 | 3461 | 3447 | 3447 | 3447 | 3447 | 3383 | 3349 | 3349 | 3349 | 3338 | 3338 | 3324 | 3324 | 3324 | | | | |
| | GA-Mce4 | 3546 | 3461 | 3461 | 3447 | 3447 | 3447 | 3447 | 3383 | 3349 | 3349 | 3349 | 3349 | 3349 | 3349 | 3348 | 3348 | | | | |
| 5a | GA | 3685 | 2948 | 2848 | 2825 | 2812 | 2782 | 2781 | 2714 | 2689 | 2684 | 2660 | 2656 | 2650 | 2650 | 2637 | 2636 | 2621 | 2621 | 2618 | 2603 |
| (5/25/10) | GA-LS | 2948 | 2848 | 2825 | 2812 | 2730 | 2702 | 2679 | 2670 | 2655 | 2634 | 2620 | 2603 | 2603 | 2603 | 2603 | 2603 | 2601 | 2599 | 2599 | |
| | GA-Mce1 | 2948 | 2848 | 2825 | 2788 | 2753 | 2684 | 2662 | 2661 | 2638 | 2623 | 2612 | 2592 | 2569 | 2554 | 2540 | 2540 | | | | |
| | GA-Mce4 | 2948 | 2848 | 2825 | 2770 | 2728 | 2707 | 2690 | 2689 | 2687 | 2672 | 2672 | 2641 | 2609 | 2597 | 2597 | 2597 | | | | |
| 6a | GA | 3524 | 2641 | 2568 | 2526 | 2480 | 2478 | 2459 | 2429 | 2426 | 2423 | 2423 | 2423 | 2423 | 2408 | 2395 | 2368 | 2359 | 2349 | 2346 | 2346 |
| (5/25/10) | GA-LS | 2641 | 2568 | 2526 | 2480 | 2477 | 2427 | 2426 | 2400 | 2376 | 2364 | 2363 | 2356 | 2343 | 2343 | 2343 | 2343 | 2343 | 2323 | 2323 | |
| | GA-Mce1 | 2641 | 2568 | 2526 | 2480 | 2477 | 2455 | 2436 | 2386 | 2356 | 2340 | 2340 | 2325 | 2325 | 2299 | 2299 | 2299 | 2299 | 2288 | | |
| | GA-Mce4 | 2641 | 2568 | 2526 | 2480 | 2477 | 2458 | 2429 | 2425 | 2424 | 2389 | 2361 | 2344 | 2336 | 2325 | 2320 | 2312 | 2301 | 2300 | | |
| 7a | GA | 4008 | 3162 | 3133 | 3110 | 3103 | 3059 | 3009 | 2926 | 2898 | 2880 | 2880 | 2877 | 2877 | 2877 | 2877 | 2877 | 2877 | 2877 | 2877 | 2877 |
| (8/25/15) | GA-LS | 3162 | 3090 | 3002 | 2930 | 2930 | 2930 | 2930 | 2922 | 2886 | 2859 | 2854 | 2843 | 2843 | 2843 | 2840 | 2840 | 2840 | 2840 | 2840 | |
| | GA-Mce1 | 3162 | 3090 | 2919 | 2889 | 2889 | 2889 | 2889 | 2889 | 2848 | 2848 | 2840 | 2840 | 2836 | 2836 | 2836 | 2830 | 2830 | | | |

| Convergence des algorithmes pour une simulation pour les instances de Dauzère-Pérès (suite) | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | |
| | GA-Mce4 | 3162 | 3090 | 2952 | 2884 | 2855 | 2855 | 2848 | 2848 | 2839 | 2799 | 2799 | 2797 | 2797 | 2797 | 2796 | 2786 | 2786 | | | |
| 8a | GA | 4102 | 3097 | 2937 | 2891 | 2877 | 2861 | 2858 | 2842 | 2826 | 2805 | 2786 | 2766 | 2748 | 2748 | 2748 | 2748 | 2712 | 2707 | 2707 | 2707 |
| (8/25/15) | GA-LS | 3097 | 2937 | 2891 | 2874 | 2861 | 2783 | 2766 | 2757 | 2753 | 2753 | 2731 | 2726 | 2710 | 2693 | 2660 | 2645 | 2626 | 2616 | | |
| | GA-Mce1 | 3097 | 2937 | 2891 | 2874 | 2828 | 2784 | 2742 | 2714 | 2696 | 2674 | 2667 | 2648 | 2648 | 2646 | 2641 | 2640 | 2608 | 2608 | | |
| | GA-Mce4 | 3097 | 2937 | 2891 | 2874 | 2861 | 2820 | 2807 | 2800 | 2792 | 2780 | 2747 | 2747 | 2737 | 2699 | 2689 | 2660 | 2622 | | | |
| 9a | GA | 4414 | 3558 | 3372 | 3290 | 3251 | 3150 | 3118 | 3083 | 3055 | 3029 | 3015 | 3003 | 2999 | 2991 | 2991 | 2985 | 2968 | 2957 | 2946 | 2945 |
| (8/25/15) | GA-LS | 3558 | 3372 | 3290 | 3251 | 3150 | 3118 | 3083 | 3042 | 3019 | 3017 | 3017 | 3009 | 2999 | 2991 | 2974 | 2960 | 2946 | 2943 | 2943 | |
| | GA-Mce1 | 3558 | 3372 | 3290 | 3251 | 3150 | 3118 | 3083 | 3042 | 3019 | 3017 | 2992 | 2982 | 2932 | 2916 | 2914 | 2908 | 2894 | | | |
| | GA-Mce4 | 3558 | 3372 | 3290 | 3251 | 3150 | 3118 | 3083 | 3042 | 3015 | 3012 | 2992 | 2976 | 2892 | 2879 | 2869 | 2862 | 2853 | | | |
| 10a | GA | 4190 | 3308 | 3138 | 3123 | 3040 | 2980 | 2980 | 2962 | 2956 | 2956 | 2954 | 2924 | 2924 | 2918 | 2918 | 2900 | 2900 | 2895 | 2889 | 2889 |
| (8/25/15) | GA-LS | 3309 | 3138 | 3100 | 3031 | 3006 | 2983 | 2980 | 2974 | 2951 | 2899 | 2892 | 2892 | 2883 | 2883 | 2840 | 2840 | 2840 | | | |
| | GA-Mce1 | 3309 | 3138 | 3100 | 2958 | 2953 | 2953 | 2943 | 2914 | 2895 | 2890 | 2890 | 2886 | 2879 | 2860 | 2820 | 2811 | 2766 | | | |
| | GA-Mce4 | 3309 | 3138 | 3100 | 3045 | 3045 | 3045 | 3045 | 2997 | 2961 | 2871 | 2871 | 2871 | 2871 | 2821 | 2819 | 2816 | | | | |
| 11a | GA | 4160 | 3126 | 3007 | 2949 | 2838 | 2790 | 2709 | 2662 | 2632 | 2598 | 2585 | 2575 | 2575 | 2569 | 2561 | 2553 | 2553 | 2553 | 2553 | 2553 |
| (8/25/15) | GA-LS | 3126 | 3007 | 2949 | 2838 | 2790 | 2709 | 2662 | 2632 | 2598 | 2583 | 2576 | 2566 | 2560 | 2554 | 2553 | 2547 | 2547 | 2546 | | |
| | GA-Mce1 | 3126 | 3007 | 2949 | 2838 | 2790 | 2709 | 2662 | 2632 | 2598 | 2583 | 2572 | 2570 | 2560 | 2560 | 2560 | 2560 | 2558 | 2555 | | |
| | GA-Mce4 | 3126 | 3007 | 2949 | 2838 | 2790 | 2709 | 2662 | 2632 | 2598 | 2583 | 2569 | 2568 | 2559 | 2555 | 2523 | 2523 | 2522 | | | |
| 12a | GA | 4648 | 3651 | 3409 | 3207 | 3162 | 3109 | 3075 | 3075 | 3075 | 3075 | 3066 | 3044 | 3041 | 3040 | 3027 | 3015 | 2994 | 2980 | 2980 | 2980 |
| (8/25/15) | GA-LS | 3651 | 3409 | 3207 | 3162 | 3109 | 3076 | 3074 | 3041 | 3036 | 3036 | 3024 | 3001 | 2981 | 2972 | 2945 | 2919 | 2911 | 2911 | 2901 | |
| | GA-Mce1 | 3651 | 3409 | 3207 | 3162 | 3109 | 3076 | 3074 | 3042 | 3024 | 3019 | 2968 | 2935 | 2924 | 2887 | 2871 | 2845 | 2843 | | | |
| | GA-Mce4 | 3651 | 3409 | 3207 | 3162 | 3109 | 3076 | 3074 | 3074 | 3060 | 3029 | 2998 | 2979 | 2949 | 2927 | 2909 | 2903 | 2894 | | | |
| 13a | GA | 5196 | 4389 | 4243 | 4124 | 4103 | 4036 | 4005 | 4003 | 3964 | 3947 | 3912 | 3883 | 3858 | 3818 | 3817 | 3811 | 3810 | 3810 | 3770 | 3760 |
| (8/25/15) | GA-LS | 4402 | 4243 | 4124 | 4081 | 3976 | 3968 | 3914 | 3911 | 3869 | 3833 | 3818 | 3800 | 3785 | 3743 | 3738 | 3723 | 3723 | 3723 | 3723 | |
| | GA-Mce1 | 4402 | 4243 | 4124 | 4081 | 3952 | 3945 | 3834 | 3825 | 3786 | 3757 | 3757 | 3757 | 3757 | 3752 | 3734 | 3723 | 3723 | 3723 | | |

| Convergence des algorithmes pour une simulation pour les instances de Dauzère-Pérès (suite) | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | |
| | GA-Mce4 | 4402 | 4243 | 4124 | 4081 | 4014 | 3971 | 3927 | 3917 | 3877 | 3858 | 3835 | 3828 | 3828 | 3828 | 3795 | 3762 | 3725 | 3711 | | |
| 14a | GA | 5506 | 4260 | 4115 | 4038 | 4017 | 3970 | 3931 | 3931 | 3925 | 3895 | 3835 | 3825 | 3797 | 3718 | 3680 | 3659 | 3659 | 3659 | 3617 | 3603 |
| (10/25/20) | GA-LS | 4170 | 4034 | 3963 | 3864 | 3803 | 3759 | 3706 | 3689 | 3684 | 3669 | 3646 | 3634 | 3613 | 3609 | 3590 | 3589 | 3576 | 3565 | 3522 | |
| | GA-Mce1 | 4170 | 4034 | 3963 | 3864 | 3803 | 3759 | 3759 | 3736 | 3705 | 3673 | 3600 | 3582 | 3544 | 3481 | 3464 | 3429 | 3429 | | | |
| | GA-Mce4 | 4170 | 4034 | 3963 | 3864 | 3803 | 3759 | 3759 | 3734 | 3707 | 3650 | 3586 | 3546 | 3489 | 3469 | 3456 | 3443 | 3421 | | | |
| 15a | GA | 5379 | 4377 | 4146 | 4050 | 3994 | 3903 | 3854 | 3842 | 3789 | 3731 | 3725 | 3670 | 3640 | 3607 | 3601 | 3600 | 3584 | 3568 | 3568 | 3558 |
| (10/25/20) | GA-LS | 3988 | 3833 | 3768 | 3622 | 3595 | 3565 | 3536 | 3511 | 3489 | 3482 | 3480 | 3480 | 3480 | 3480 | 3480 | | | | | |
| | GA-Mce1 | 3988 | 3833 | 3768 | 3622 | 3595 | 3565 | 3536 | 3511 | 3489 | 3482 | 3480 | 3480 | 3480 | 3480 | | | | | | |
| | GA-Mce4 | 3988 | 3833 | 3768 | 3622 | 3595 | 3573 | 3528 | 3495 | 3488 | 3487 | 3438 | 3420 | 3409 | 3390 | | | | | | |
| 16a | GA | 5200 | 4463 | 4233 | 4097 | 4078 | 4045 | 3953 | 3923 | 3883 | 3835 | 3811 | 3785 | 3785 | 3764 | 3741 | 3741 | 3741 | 3714 | 3714 | 3714 |
| (10/25/20) | GA-LS | 4463 | 4251 | 4097 | 4073 | 4059 | 3963 | 3925 | 3890 | 3819 | 3795 | 3748 | 3745 | 3740 | 3720 | 3710 | 3709 | 3656 | 3636 | | |
| | GA-Mce1 | 4463 | 4251 | 4097 | 4073 | 4017 | 3965 | 3855 | 3801 | 3762 | 3755 | 3712 | 3711 | 3705 | 3657 | 3606 | 3586 | 3581 | 3580 | | |
| | GA-Mce4 | 4463 | 4251 | 4097 | 4073 | 4059 | 3915 | 3847 | 3825 | 3793 | 3787 | 3763 | 3751 | 3734 | 3677 | 3666 | 3640 | 3598 | | | |
| 17a | GA | 5455 | 4664 | 4398 | 4221 | 4179 | 4057 | 3985 | 3963 | 3963 | 3960 | 3959 | 3955 | 3894 | 3850 | 3833 | 3817 | 3770 | 3732 | 3732 | 3727 |
| (10/25/20) | GA-LS | 4177 | 4071 | 3962 | 3850 | 3812 | 3801 | 3774 | 3739 | 3733 | 3703 | 3683 | 3677 | 3652 | 3638 | 3638 | 3638 | 3638 | | | |
| | GA-Mce1 | 4177 | 4071 | 3962 | 3850 | 3812 | 3778 | 3729 | 3655 | 3647 | 3567 | 3554 | 3515 | 3498 | 3498 | 3474 | 3454 | 3445 | | | |
| | GA-Mce4 | 4177 | 4071 | 3962 | 3850 | 3812 | 3756 | 3714 | 3677 | 3660 | 3589 | 3573 | 3564 | 3560 | 3557 | 3555 | 3555 | | | | |
| 18a | GA | 5493 | 4317 | 4082 | 4007 | 3911 | 3820 | 3782 | 3733 | 3733 | 3719 | 3684 | 3684 | 3655 | 3635 | 3635 | 3630 | 3626 | 3626 | 3626 | 3621 |
| (10/25/20) | GA-LS | 4305 | 4220 | 4085 | 3971 | 3946 | 3946 | 3931 | 3893 | 3884 | 3846 | 3779 | 3739 | 3721 | 3694 | 3674 | 3636 | 3622 | 3590 | 3555 | |
| | GA-Mce1 | 4305 | 4220 | 4085 | 3971 | 3946 | 3906 | 3842 | 3782 | 3745 | 3697 | 3688 | 3670 | 3632 | 3597 | 3597 | 3597 | 3597 | | | |
| | GA-Mce4 | 4305 | 4220 | 4085 | 3971 | 3946 | 3888 | 3851 | 3832 | 3748 | 3686 | 3627 | 3591 | 3570 | 3556 | 3549 | 3509 | 3508 | | | |

TABLEAU 2.8: Convergence des algorithmes pour une simulation
pour les instances d'Adams (Adams et al., 1988) avec la règle rdata

| Convergence des algorithmes pour une simulation pour les instances d'Adams avec la règle rdata (suite) | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | |
| 101 | GA | 867 | 536 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 |
| (5/5/10) | GA-LS | 536 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | |
| | GA-Mce1 | 536 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | |
| | GA-Mce4 | 536 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 533 | 531 | 529 | |
| 102 | GA | 836 | 586 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 |
| (5/5/10) | GA-LS | 586 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | |
| | GA-Mce1 | 586 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 550 | 550 | 550 | | |
| | GA-Mce4 | 586 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 569 | 565 | 558 | 558 | 558 | 558 | 558 | 549 | | | |
| 103 | GA | 678 | 448 | 424 | 424 | 421 | 419 | 418 | 418 | 418 | 418 | 418 | 418 | 418 | 418 | 418 | 418 | 418 | 418 | 418 | 418 |
| (5/5/10) | GA-LS | 448 | 424 | 424 | 424 | 424 | 424 | 424 | 421 | 417 | 417 | 417 | 417 | 417 | 417 | 417 | 417 | 417 | 417 | 417 | |
| | GA-Mce1 | 448 | 424 | 424 | 417 | 417 | 417 | 417 | 417 | 417 | 417 | 417 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | | |
| | GA-Mce4 | 448 | 424 | 424 | 424 | 424 | 424 | 424 | 419 | 417 | 417 | 417 | 417 | 417 | 417 | 417 | 404 | 404 | | | |
| 104 | GA | 769 | 505 | 505 | 505 | 505 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 |
| (5/5/10) | GA-LS | 505 | 505 | 505 | 505 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | |
| | GA-Mce1 | 505 | 505 | 497 | 487 | 487 | 487 | 482 | 482 | 482 | 482 | 482 | 482 | 482 | 482 | 482 | 482 | 482 | 482 | | |
| | GA-Mce4 | 505 | 505 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | 490 | | | |
| 105 | GA | 723 | 492 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 |
| (5/5/10) | GA-LS | 492 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 451 | |
| | GA-Mce1 | 492 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 445 | | |
| | GA-Mce4 | 492 | 467 | 467 | 467 | 467 | 467 | 467 | 467 | 451 | 451 | 451 | 451 | 445 | 434 | 434 | 434 | 434 | | | |
| 106 | GA | 1246 | 856 | 840 | 832 | 821 | 821 | 821 | 821 | 821 | 821 | 801 | 793 | 791 | 791 | 791 | 791 | 791 | 791 | 791 | 791 |

| Convergence des algorithmes pour une simulation pour les instances d'Adams (suite) | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | |
| (5/5/15) | GA-LS | 856 | 840 | 832 | 810 | 801 | 772 | 771 | 771 | 771 | 771 | 771 | 771 | 771 | 771 | 771 | 768 | 768 | 756 | | |
| | GA-Mce1 | 856 | 840 | 832 | 821 | 809 | 801 | 773 | 773 | 757 | 754 | 754 | 754 | 754 | 751 | 751 | 751 | 733 | | | |
| | GA-Mce4 | 856 | 840 | 832 | 810 | 786 | 773 | 773 | 773 | 768 | 765 | 765 | 759 | 756 | 749 | 749 | 742 | 733 | | | |
| 107 | GA | 1182 | 871 | 834 | 834 | 834 | 834 | 834 | 834 | 834 | 834 | 834 | 834 | 834 | 834 | 834 | 817 | 812 | 812 | 812 | 812 |
| (5/5/15) | GA-LS | 871 | 834 | 834 | 834 | 834 | 821 | 812 | 812 | 812 | 812 | 812 | 812 | 804 | 804 | 804 | 796 | 796 | 796 | 796 | |
| | GA-Mce1 | 871 | 834 | 825 | 818 | 807 | 796 | 796 | 796 | 796 | 796 | 796 | 796 | 796 | 796 | 796 | 796 | 796 | | | |
| | GA-Mce4 | 871 | 834 | 825 | 818 | 818 | 791 | 788 | 788 | 788 | 788 | 788 | 788 | 788 | 788 | 788 | 788 | | | | |
| 108 | GA | 996 | 696 | 661 | 654 | 625 | 621 | 621 | 621 | 614 | 612 | 610 | 605 | 605 | 605 | 605 | 605 | 605 | 605 | 605 | 605 |
| (5/5/15) | GA-LS | 696 | 661 | 654 | 625 | 621 | 620 | 610 | 610 | 605 | 605 | 601 | 595 | 595 | 595 | 595 | 595 | 595 | 595 | | |
| | GA-Mce1 | 696 | 661 | 654 | 625 | 621 | 605 | 605 | 605 | 605 | 603 | 603 | 603 | 603 | 583 | 582 | 582 | 582 | | | |
| | GA-Mce4 | 696 | 661 | 654 | 625 | 621 | 614 | 608 | 605 | 605 | 605 | 605 | 605 | 601 | 594 | 588 | 571 | 558 | | | |
| 109 | GA | 1273 | 757 | 737 | 737 | 737 | 737 | 735 | 724 | 724 | 724 | 724 | 724 | 724 | 724 | 724 | 724 | 724 | 724 | 724 | 724 |
| (5/5/15) | GA-LS | 757 | 737 | 737 | 737 | 737 | 737 | 724 | 724 | 724 | 724 | 724 | 724 | 724 | 724 | 724 | 724 | 724 | 724 | 724 | |
| | GA-Mce1 | 757 | 737 | 737 | 737 | 737 | 737 | 724 | 724 | 724 | 724 | 724 | 724 | 724 | 724 | 724 | 724 | 724 | | | |
| | GA-Mce4 | 757 | 737 | 737 | 737 | 737 | 737 | 737 | 737 | 737 | 737 | 737 | 737 | 737 | 737 | 716 | 716 | 716 | | | |
| 110 | GA | 1142 | 740 | 736 | 695 | 687 | 687 | 687 | 687 | 687 | 687 | 687 | 687 | 687 | 687 | 687 | 687 | 687 | 687 | 687 | 687 |
| (5/5/15) | GA-LS | 740 | 736 | 728 | 728 | 721 | 721 | 711 | 704 | 697 | 687 | 687 | 687 | 687 | 687 | 687 | 687 | 687 | | | |
| | GA-Mce1 | 740 | 736 | 728 | 721 | 721 | 718 | 715 | 715 | 698 | 697 | 688 | 688 | 688 | 688 | 686 | 685 | 680 | | | |
| | GA-Mce4 | 740 | 736 | 692 | 687 | 687 | 687 | 687 | 687 | 687 | 687 | 687 | 687 | 687 | 687 | 687 | 683 | 680 | | | |
| 111 | GA | 1596 | 1068 | 1050 | 1050 | 1050 | 1048 | 1035 | 1031 | 1031 | 1031 | 1031 | 1031 | 1023 | 1023 | 1021 | 1021 | 1020 | 1020 | 1020 | 1020 |
| (5/5/20) | GA-LS | 1068 | 1050 | 1049 | 1049 | 1035 | 1035 | 1031 | 1020 | 1017 | 1007 | 1007 | 1007 | 1007 | 1007 | 1007 | 1007 | 1007 | 1007 | | |
| | GA-Mce1 | 1068 | 1050 | 1049 | 1027 | 1023 | 1023 | 1023 | 1023 | 1023 | 1023 | 1016 | 1007 | 1007 | 1007 | 1007 | 1007 | | | | |
| | GA-Mce4 | 1068 | 1050 | 1049 | 1049 | 1049 | 1034 | 1007 | 1007 | 1003 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | | | | | |
| 112 | GA | 1587 | 1143 | 1119 | 1093 | 1092 | 1092 | 1092 | 1092 | 1092 | 1092 | 1092 | 1092 | 1092 | 1092 | 1092 | 1092 | 1092 | 1092 | 1092 | 1092 |

| Convergence des algorithmes pour une simulation pour les instances d'Adams (suite) | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | |
| (5/5/20) | GA-LS | 1158 | 1123 | 1103 | 1103 | 1103 | 1103 | 1085 | 1085 | 1068 | 1068 | 1068 | 1068 | 1068 | 1068 | 1068 | 1068 | 1068 | | | |
| | GA-Mce1 | 1158 | 1123 | 1103 | 1103 | 1103 | 1103 | 1103 | 1088 | 1088 | 1084 | 1084 | 1084 | 1084 | 1084 | 1084 | 1066 | 1066 | | | |
| | GA-Mce4 | 1158 | 1123 | 1103 | 1103 | 1103 | 1085 | 1085 | 1085 | 1080 | 1051 | 1049 | 1049 | 1049 | 1045 | 1045 | 1045 | 1037 | | | |
| 113 | GA | 1361 | 893 | 874 | 860 | 835 | 822 | 818 | 818 | 818 | 816 | 810 | 810 | 810 | 810 | 810 | 810 | 810 | 810 | 810 | 810 |
| (5/5/20) | GA-LS | 893 | 874 | 860 | 843 | 822 | 818 | 818 | 818 | 818 | 816 | 816 | 816 | 816 | 816 | 808 | 801 | 796 | 795 | 788 | |
| | GA-Mce1 | 893 | 874 | 860 | 843 | 822 | 818 | 816 | 815 | 809 | 800 | 797 | 787 | 782 | 782 | 782 | 782 | 782 | | | |
| | GA-Mce4 | 893 | 874 | 860 | 843 | 822 | 818 | 817 | 817 | 816 | 805 | 796 | 796 | 796 | 770 | 765 | 763 | 763 | | | |
| 114 | GA | 1528 | 1016 | 976 | 944 | 940 | 925 | 925 | 925 | 925 | 925 | 925 | 914 | 905 | 905 | 905 | 904 | 904 | 904 | 904 | 904 |
| (5/5/20) | GA-LS | 1016 | 976 | 944 | 940 | 925 | 925 | 925 | 925 | 925 | 913 | 911 | 904 | 904 | 904 | 904 | 904 | 904 | 904 | | |
| | GA-Mce1 | 1016 | 976 | 944 | 925 | 925 | 911 | 904 | 904 | 893 | 889 | 889 | 889 | 889 | 889 | 889 | 889 | 889 | | | |
| | GA-Mce4 | 1016 | 976 | 944 | 925 | 911 | 910 | 910 | 910 | 910 | 910 | 898 | 889 | 889 | 889 | 889 | 889 | 889 | | | |
| 115 | GA | 1270 | 936 | 928 | 917 | 875 | 870 | 869 | 865 | 865 | 865 | 865 | 865 | 865 | 865 | 865 | 865 | 865 | 865 | 865 | 861 |
| (5/5/20) | GA-LS | 936 | 928 | 928 | 884 | 874 | 874 | 869 | 869 | 866 | 866 | 866 | 861 | 861 | 861 | 861 | 861 | 845 | 842 | | |
| | GA-Mce1 | 936 | 928 | 928 | 928 | 928 | 928 | 897 | 878 | 865 | 865 | 865 | 865 | 865 | 865 | 865 | 865 | 865 | | | |
| | GA-Mce4 | 936 | 928 | 928 | 928 | 928 | 928 | 928 | 904 | 855 | 847 | 847 | 847 | 834 | 832 | 832 | 832 | 832 | | | |
| 116 | GA | 1356 | 1012 | 960 | 953 | 937 | 927 | 926 | 901 | 889 | 889 | 889 | 880 | 878 | 878 | 878 | 878 | 878 | 878 | 878 | 878 |
| (10/10/10) | GA-LS | 1012 | 960 | 953 | 923 | 923 | 923 | 901 | 901 | 901 | 901 | 898 | 898 | 886 | 886 | 886 | 886 | 886 | 885 | 878 | |
| | GA-Mce1 | 1012 | 960 | 953 | 929 | 926 | 923 | 923 | 898 | 898 | 898 | 898 | 885 | 878 | 872 | 872 | 855 | 854 | 843 | | |
| | GA-Mce4 | 1012 | 960 | 953 | 926 | 923 | 908 | 898 | 898 | 898 | 878 | 867 | 860 | 860 | 850 | 826 | 820 | 816 | | | |
| 117 | GA | 1311 | 938 | 932 | 886 | 885 | 872 | 862 | 862 | 862 | 862 | 856 | 856 | 856 | 856 | 856 | 856 | 856 | 856 | 856 | 856 |
| (10/10/10) | GA-LS | 938 | 911 | 880 | 867 | 863 | 862 | 862 | 862 | 862 | 862 | 862 | 862 | 862 | 862 | 862 | 862 | 862 | 862 | 852 | |
| | GA-Mce1 | 938 | 911 | 856 | 847 | 846 | 846 | 845 | 845 | 845 | 845 | 845 | 845 | 845 | 845 | 845 | 845 | 845 | | | |
| | GA-Mce4 | 938 | 911 | 856 | 847 | 843 | 838 | 836 | 836 | 836 | 836 | 836 | 836 | 836 | 836 | 836 | 836 | 836 | | | |
| 118 | GA | 1244 | 760 | 704 | 696 | 693 | 693 | 687 | 670 | 658 | 653 | 653 | 653 | 653 | 653 | 653 | 653 | 653 | 653 | 653 | 653 |

| Convergence des algorithmes pour une simulation pour les instances d'Adams (suite) | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | |
| (10/10/10) | GA-LS | 760 | 704 | 696 | 693 | 693 | 693 | 687 | 655 | 653 | 653 | 653 | 653 | 651 | 651 | 645 | 645 | 645 | 643 | | |
| | GA-Mce1 | 760 | 704 | 696 | 693 | 693 | 693 | 693 | 655 | 655 | 655 | 655 | 653 | 651 | 651 | 645 | 645 | 645 | 645 | | |
| | GA-Mce4 | 760 | 704 | 696 | 693 | 691 | 684 | 667 | 664 | 655 | 653 | 646 | 646 | 646 | 646 | 645 | 638 | 638 | | | |
| 119 | GA | 1303 | 841 | 823 | 817 | 817 | 771 | 766 | 760 | 759 | 756 | 751 | 751 | 751 | 739 | 728 | 728 | 710 | 699 | 698 | 698 |
| (10/10/10) | GA-LS | 841 | 823 | 817 | 776 | 731 | 731 | 721 | 714 | 707 | 702 | 702 | 702 | 702 | 702 | 690 | 690 | 680 | 680 | 678 | |
| | GA-Mce1 | 841 | 823 | 817 | 805 | 775 | 752 | 738 | 736 | 714 | 714 | 714 | 706 | 696 | 686 | 686 | 686 | 686 | 686 | 686 | |
| | GA-Mce4 | 841 | 823 | 817 | 796 | 790 | 776 | 760 | 746 | 727 | 716 | 713 | 711 | 690 | 679 | 679 | 679 | 677 | 677 | | |
| 120 | GA | 1371 | 928 | 889 | 886 | 874 | 874 | 872 | 855 | 855 | 849 | 848 | 844 | 836 | 808 | 808 | 808 | 804 | 804 | 798 | 798 |
| (10/10/10) | GA-LS | 928 | 889 | 885 | 861 | 861 | 861 | 850 | 850 | 850 | 839 | 835 | 835 | 813 | 813 | 795 | 794 | 787 | 785 | 785 | |
| | GA-Mce1 | 928 | 889 | 885 | 871 | 861 | 861 | 850 | 845 | 844 | 844 | 844 | 830 | 816 | 804 | 800 | 794 | 768 | 757 | | |
| | GA-Mce4 | 928 | 889 | 874 | 871 | 846 | 836 | 821 | 812 | 811 | 798 | 798 | 790 | 787 | 784 | 784 | 766 | 765 | 765 | | |
| 121 | GA | 2076 | 1357 | 1291 | 1265 | 1216 | 1200 | 1200 | 1200 | 1177 | 1176 | 1154 | 1154 | 1148 | 1148 | 1134 | 1134 | 1134 | 1134 | 1134 | 1134 |
| (10/10/15) | GA-LS | 1348 | 1303 | 1255 | 1235 | 1224 | 1217 | 1217 | 1217 | 1170 | 1170 | 1167 | 1158 | 1152 | 1140 | 1140 | 1117 | 1117 | 1117 | | |
| | GA-Mce1 | 1357 | 1291 | 1265 | 1216 | 1200 | 1186 | 1186 | 1186 | 1186 | 1186 | 1186 | 1186 | 1163 | 1157 | 1157 | 1150 | 1150 | | | |
| | GA-Mce4 | 1357 | 1291 | 1265 | 1216 | 1200 | 1186 | 1186 | 1186 | 1168 | 1168 | 1160 | 1160 | 1148 | 1125 | 1119 | 1119 | 1116 | | | |
| 122 | GA | 1858 | 1298 | 1268 | 1263 | 1260 | 1246 | 1225 | 1220 | 1220 | 1220 | 1220 | 1220 | 1220 | 1174 | 1174 | 1174 | 1174 | 1159 | 1159 | 1159 |
| (10/10/15) | GA-LS | 1262 | 1256 | 1244 | 1230 | 1230 | 1225 | 1225 | 1225 | 1225 | 1225 | 1225 | 1164 | 1164 | 1164 | 1159 | 1159 | 1159 | 1159 | | |
| | GA-Mce1 | 1298 | 1268 | 1263 | 1262 | 1207 | 1175 | 1171 | 1159 | 1159 | 1159 | 1159 | 1159 | 1158 | 1143 | 1123 | 1110 | 1098 | 1098 | | |
| | GA-Mce4 | 1298 | 1268 | 1263 | 1258 | 1255 | 1246 | 1238 | 1238 | 1224 | 1218 | 1215 | 1200 | 1184 | 1168 | 1154 | 1148 | 1106 | 1081 | | |
| 123 | GA | 1608 | 1050 | 1034 | 1026 | 1019 | 999 | 977 | 966 | 961 | 961 | 961 | 940 | 940 | 934 | 934 | 934 | 924 | 911 | 896 | 890 |
| (10/10/15) | GA-LS | 1050 | 1048 | 1028 | 1003 | 987 | 983 | 952 | 947 | 947 | 930 | 912 | 904 | 895 | 890 | 877 | 868 | 865 | 861 | | |
| | GA-Mce1 | 1050 | 1034 | 1017 | 1005 | 983 | 972 | 954 | 951 | 931 | 923 | 899 | 883 | 869 | 860 | 858 | 857 | 857 | 857 | | |
| | GA-Mce4 | 1050 | 1034 | 1017 | 978 | 954 | 928 | 924 | 908 | 897 | 881 | 881 | 867 | 854 | 854 | 851 | 850 | 847 | | | |
| 124 | GA | 1687 | 1204 | 1101 | 1072 | 1050 | 1022 | 1017 | 1017 | 1016 | 1008 | 1008 | 1001 | 999 | 998 | 990 | 990 | 990 | 978 | 978 | 978 |

| Convergence des algorithmes pour une simulation pour les instances d'Adams (suite) | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | |
| (10/10/15) | GA-LS | 1212 | 1131 | 1101 | 1076 | 1059 | 1026 | 1022 | 1017 | 992 | 989 | 989 | 980 | 978 | 974 | 974 | 974 | 974 | 974 | 974 | |
| | GA-Mce1 | 1213 | 1101 | 1072 | 1050 | 1022 | 1017 | 1011 | 1011 | 1002 | 1000 | 980 | 975 | 974 | 970 | 969 | 965 | 960 | 951 | | |
| | GA-Mce4 | 1213 | 1101 | 1072 | 1050 | 1022 | 1017 | 1017 | 1016 | 1006 | 1005 | 1000 | 987 | 981 | 969 | 967 | 963 | 963 | | | |
| 125 | GA | 1736 | 1231 | 1176 | 1163 | 1146 | 1133 | 1133 | 1114 | 1114 | 1114 | 1087 | 1076 | 1076 | 1076 | 1076 | 1076 | 1076 | 1076 | 1076 | 1076 |
| (10/10/15) | GA-LS | 1301 | 1269 | 1196 | 1176 | 1120 | 1120 | 1096 | 1094 | 1083 | 1079 | 1079 | 1079 | 1071 | 1071 | 1071 | 1071 | 1071 | 1071 | | |
| | GA-Mce1 | 1301 | 1247 | 1216 | 1195 | 1152 | 1146 | 1138 | 1134 | 1103 | 1095 | 1095 | 1095 | 1085 | 1069 | 1069 | 1069 | 1069 | | | |
| | GA-Mce4 | 1231 | 1176 | 1163 | 1146 | 1133 | 1131 | 1119 | 1114 | 1114 | 1114 | 1090 | 1076 | 1076 | 1076 | 1064 | 1061 | 1061 | | | |
| 126 | GA | 2443 | 1655 | 1649 | 1643 | 1607 | 1560 | 1559 | 1559 | 1559 | 1559 | 1559 | 1559 | 1559 | 1554 | 1534 | 1531 | 1523 | 1520 | 1516 | 1516 |
| (10/10/20) | GA-LS | 1661 | 1620 | 1565 | 1543 | 1543 | 1525 | 1525 | 1515 | 1507 | 1507 | 1488 | 1486 | 1486 | 1486 | 1486 | 1486 | 1486 | 1486 | | |
| | GA-Mce1 | 1655 | 1649 | 1635 | 1596 | 1586 | 1577 | 1574 | 1574 | 1574 | 1574 | 1574 | 1574 | 1565 | 1538 | 1517 | 1501 | 1493 | | | |
| | GA-Mce4 | 1655 | 1649 | 1601 | 1591 | 1576 | 1553 | 1527 | 1521 | 1521 | 1521 | 1513 | 1509 | 1495 | 1492 | 1483 | 1483 | 1447 | | | |
| 127 | GA | 2261 | 1621 | 1588 | 1588 | 1572 | 1526 | 1474 | 1473 | 1463 | 1453 | 1448 | 1448 | 1448 | 1448 | 1448 | 1448 | 1437 | 1434 | 1431 | 1430 |
| (10/10/20) | GA-LS | 1609 | 1572 | 1533 | 1513 | 1500 | 1489 | 1489 | 1425 | 1425 | 1417 | 1375 | 1373 | 1373 | 1373 | 1357 | 1347 | 1347 | 1347 | | |
| | GA-Mce1 | 1621 | 1588 | 1561 | 1516 | 1516 | 1476 | 1474 | 1474 | 1473 | 1458 | 1445 | 1445 | 1445 | 1424 | 1379 | 1360 | 1354 | 1343 | | |
| | GA-Mce4 | 1621 | 1588 | 1561 | 1532 | 1498 | 1483 | 1471 | 1465 | 1464 | 1464 | 1436 | 1413 | 1405 | 1389 | 1388 | 1388 | 1388 | 1369 | | |
| 128 | GA | 2090 | 1446 | 1379 | 1366 | 1330 | 1282 | 1266 | 1244 | 1217 | 1216 | 1211 | 1206 | 1203 | 1198 | 1193 | 1182 | 1178 | 1178 | 1178 | 1178 |
| (10/10/20) | GA-LS | 1443 | 1413 | 1341 | 1308 | 1293 | 1265 | 1250 | 1232 | 1221 | 1220 | 1197 | 1181 | 1179 | 1178 | 1170 | 1162 | 1153 | 1152 | | |
| | GA-Mce1 | 1446 | 1379 | 1364 | 1351 | 1310 | 1281 | 1251 | 1203 | 1191 | 1175 | 1161 | 1140 | 1136 | 1125 | 1113 | 1087 | 1087 | 1087 | | |
| | GA-Mce4 | 1446 | 1379 | 1364 | 1334 | 1288 | 1276 | 1266 | 1261 | 1253 | 1233 | 1220 | 1201 | 1173 | 1147 | 1134 | 1133 | 1129 | 1127 | | |
| 129 | GA | 2103 | 1585 | 1529 | 1436 | 1420 | 1420 | 1366 | 1331 | 1328 | 1318 | 1274 | 1260 | 1256 | 1256 | 1252 | 1251 | 1251 | 1249 | 1245 | 1245 |
| (10/10/20) | GA-LS | 1405 | 1330 | 1315 | 1289 | 1269 | 1269 | 1256 | 1216 | 1193 | 1187 | 1153 | 1128 | 1111 | 1111 | 1110 | 1102 | 1102 | 1102 | | |
| | GA-Mce1 | 1405 | 1330 | 1315 | 1271 | 1264 | 1253 | 1240 | 1221 | 1204 | 1172 | 1158 | 1151 | 1145 | 1132 | 1122 | 1117 | 1103 | 1093 | | |
| | GA-Mce4 | 1405 | 1330 | 1315 | 1266 | 1256 | 1247 | 1244 | 1225 | 1225 | 1222 | 1211 | 1198 | 1198 | 1196 | 1196 | 1173 | 1165 | | | |
| 130 | GA | 2230 | 1677 | 1608 | 1567 | 1530 | 1526 | 1526 | 1496 | 1478 | 1476 | 1476 | 1467 | 1436 | 1434 | 1426 | 1426 | 1426 | 1426 | 1423 | 1422 |

| Convergence des algorithmes pour une simulation pour les instances d'Adams (suite) | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | |
| (10/10/20) | GA-LS | 1665 | 1586 | 1557 | 1524 | 1509 | 1474 | 1462 | 1427 | 1397 | 1397 | 1397 | 1379 | 1375 | 1369 | 1365 | 1360 | 1360 | 1360 | | |
| | GA-Mce1 | 1677 | 1608 | 1567 | 1530 | 1526 | 1508 | 1495 | 1460 | 1460 | 1439 | 1402 | 1395 | 1395 | 1395 | 1383 | 1362 | 1362 | 1362 | | |
| | GA-Mce4 | 1677 | 1608 | 1567 | 1530 | 1526 | 1506 | 1490 | 1461 | 1446 | 1445 | 1427 | 1412 | 1410 | 1365 | 1354 | 1336 | 1329 | 1329 | | |
| 131 | GA | 3426 | 2493 | 2441 | 2414 | 2372 | 2274 | 2207 | 2161 | 2130 | 2110 | 2110 | 2110 | 2100 | 2092 | 2015 | 2011 | 2011 | 2005 | 2004 | 2004 |
| (10/10/30) | GA-LS | 1870 | 1804 | 1804 | 1804 | 1755 | 1739 | 1723 | 1711 | 1711 | 1711 | 1711 | 1711 | 1711 | 1711 | 1711 | 1711 | | | | |
| | GA-Mce1 | 1870 | 1804 | 1761 | 1737 | 1729 | 1724 | 1710 | 1703 | 1691 | 1690 | 1689 | 1689 | 1689 | 1689 | 1674 | 1674 | | | | |
| | GA-Mce4 | 1870 | 1804 | 1761 | 1737 | 1726 | 1720 | 1716 | 1712 | 1710 | 1698 | 1694 | 1681 | 1681 | 1668 | 1666 | 1666 | | | | |
| 132 | GA | 3248 | 2443 | 2301 | 2246 | 2222 | 2199 | 2195 | 2185 | 2164 | 2162 | 2162 | 2162 | 2097 | 2097 | 2097 | 2097 | 2097 | 2096 | 2092 | 2092 |
| (10/10/30) | GA-LS | 2275 | 2275 | 2076 | 2071 | 2061 | 2031 | 2031 | 2026 | 2026 | 2026 | 2026 | 2026 | 2026 | 2026 | 2026 | 2026 | 2026 | 2026 | | |
| | GA-Mce1 | 2275 | 2202 | 2102 | 2079 | 2061 | 2061 | 2031 | 2031 | 2031 | 2026 | 2026 | 2026 | 2026 | 2026 | 2026 | 2026 | 2026 | | | |
| | GA-Mce4 | 2275 | 2202 | 2102 | 2079 | 2061 | 2061 | 2031 | 2026 | 2026 | 2021 | 2021 | 2021 | 2021 | 2021 | 2021 | 2021 | | | | |
| 133 | GA | 2876 | 2142 | 2113 | 2091 | 2055 | 2012 | 1996 | 1979 | 1974 | 1969 | 1957 | 1933 | 1933 | 1896 | 1880 | 1853 | 1834 | 1834 | 1829 | 1802 |
| (10/10/30) | GA-LS | 1556 | 1456 | 1429 | 1429 | 1406 | 1380 | 1351 | 1318 | 1317 | 1317 | 1310 | 1307 | 1307 | 1307 | 1307 | 1307 | 1302 | 1298 | | |
| | GA-Mce1 | 1559 | 1489 | 1462 | 1432 | 1345 | 1331 | 1331 | 1329 | 1323 | 1307 | 1291 | 1288 | 1288 | 1277 | 1277 | 1277 | 1277 | | | |
| | GA-Mce4 | 1559 | 1489 | 1462 | 1432 | 1345 | 1331 | 1331 | 1307 | 1305 | 1291 | 1278 | 1278 | 1278 | 1276 | 1268 | 1268 | 1261 | | | |
| 134 | GA | 2989 | 2254 | 2155 | 2093 | 2037 | 1937 | 1937 | 1930 | 1887 | 1869 | 1838 | 1828 | 1813 | 1808 | 1792 | 1772 | 1751 | 1751 | 1750 | 1750 |
| (10/10/30) | GA-LS | 1733 | 1696 | 1657 | 1641 | 1612 | 1600 | 1586 | 1585 | 1555 | 1551 | 1551 | 1540 | 1540 | 1540 | 1540 | 1540 | 1540 | | | |
| | GA-Mce1 | 1733 | 1676 | 1657 | 1602 | 1575 | 1574 | 1571 | 1568 | 1550 | 1545 | 1545 | 1537 | 1527 | 1527 | 1527 | 1527 | | | | |
| | GA-Mce4 | 1733 | 1676 | 1657 | 1620 | 1583 | 1552 | 1536 | 1535 | 1522 | 1518 | 1518 | 1504 | 1504 | 1504 | 1495 | 1495 | | | | |
| 135 | GA | 3291 | 2350 | 2288 | 2249 | 2246 | 2223 | 2187 | 2187 | 2173 | 2163 | 2155 | 2124 | 2107 | 2069 | 2069 | 2069 | 2069 | 2047 | 2047 | 2035 |
| (10/10/30) | GA-LS | 2087 | 1960 | 1931 | 1909 | 1887 | 1887 | 1870 | 1870 | 1870 | 1870 | 1870 | 1870 | 1853 | 1836 | 1836 | 1836 | 1836 | | | |
| | GA-Mce1 | 2094 | 1990 | 1981 | 1961 | 1915 | 1900 | 1856 | 1825 | 1798 | 1798 | 1798 | 1798 | 1780 | 1776 | 1776 | 1776 | | | | |
| | GA-Mce4 | 2094 | 1990 | 1981 | 1930 | 1922 | 1894 | 1847 | 1831 | 1817 | 1810 | 1788 | 1766 | 1766 | 1766 | 1766 | | | | | |
| 136 | GA | 2012 | 1473 | 1448 | 1402 | 1388 | 1370 | 1370 | 1342 | 1341 | 1340 | 1333 | 1317 | 1313 | 1309 | 1303 | 1302 | 1302 | 1302 | 1302 | 1292 |

| Convergence des algorithmes pour une simulation pour les instances d'Adams (suite) | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | |
| (15/15/15) | GA-LS | 1145 | 1092 | 1086 | 1074 | 1042 | 1037 | 1037 | 1024 | 1024 | 1024 | 1017 | 1001 | 995 | 995 | 995 | 995 | 995 | | | |
| | GA-Mce1 | 1131 | 1119 | 1022 | 1001 | 987 | 981 | 968 | 968 | 967 | 967 | 964 | 964 | 964 | 964 | 964 | 956 | 954 | | | |
| | GA-Mce4 | 1131 | 1119 | 1018 | 1012 | 1012 | 1000 | 990 | 990 | 989 | 980 | 978 | 978 | 971 | 965 | 953 | 950 | 950 | | | |
| 137 | GA | 2236 | 1746 | 1645 | 1570 | 1528 | 1523 | 1486 | 1466 | 1457 | 1454 | 1447 | 1447 | 1447 | 1443 | 1438 | 1437 | 1420 | 1418 | 1408 | 1406 |
| (15/15/15) | GA-LS | 1379 | 1372 | 1372 | 1372 | 1356 | 1320 | 1320 | 1320 | 1307 | 1307 | 1307 | 1307 | 1307 | 1307 | 1306 | 1306 | 1306 | 1306 | 1306 | |
| | GA-Mce1 | 1411 | 1342 | 1323 | 1314 | 1314 | 1310 | 1307 | 1307 | 1306 | 1306 | 1306 | 1306 | 1306 | 1301 | 1301 | 1284 | 1276 | | | |
| | GA-Mce4 | 1411 | 1342 | 1323 | 1314 | 1314 | 1314 | 1307 | 1307 | 1282 | 1282 | 1282 | 1281 | 1269 | 1269 | 1269 | 1264 | | | | |
| 138 | GA | 2220 | 1567 | 1492 | 1389 | 1364 | 1341 | 1341 | 1341 | 1336 | 1293 | 1255 | 1252 | 1252 | 1252 | 1252 | 1252 | 1252 | 1252 | 1252 | 1252 |
| (15/15/15) | GA-LS | 1278 | 1132 | 1097 | 1097 | 1097 | 1097 | 1097 | 1097 | 1097 | 1097 | 1089 | 1089 | 1084 | 1084 | 1084 | 1084 | 1084 | 1084 | 1084 | |
| | GA-Mce1 | 1278 | 1132 | 1109 | 1086 | 1084 | 1084 | 1062 | 1057 | 1057 | 1057 | 1057 | 1057 | 1057 | 1057 | 1057 | 1057 | 1057 | | | |
| | GA-Mce4 | 1278 | 1132 | 1109 | 1086 | 1084 | 1084 | 1084 | 1084 | 1079 | 1065 | 1051 | 1051 | 1051 | 1051 | 1051 | 1051 | 1051 | | | |
| 139 | GA | 2279 | 1587 | 1540 | 1454 | 1418 | 1356 | 1345 | 1330 | 1297 | 1251 | 1251 | 1240 | 1239 | 1239 | 1232 | 1220 | 1216 | 1216 | 1216 | 1210 |
| (15/15/15) | GA-LS | 1200 | 1160 | 1153 | 1148 | 1122 | 1116 | 1097 | 1097 | 1090 | 1086 | 1086 | 1086 | 1086 | 1086 | 1083 | 1078 | 1078 | | | |
| | GA-Mce1 | 1200 | 1165 | 1134 | 1118 | 1105 | 1097 | 1096 | 1078 | 1076 | 1072 | 1066 | 1066 | 1066 | 1066 | 1066 | 1066 | | | | |
| | GA-Mce4 | 1200 | 1165 | 1134 | 1118 | 1105 | 1097 | 1085 | 1079 | 1054 | 1054 | 1053 | 1034 | 1032 | 1032 | 1032 | 1032 | | | | |
| 140 | GA | 2016 | 1614 | 1537 | 1517 | 1496 | 1449 | 1406 | 1395 | 1383 | 1382 | 1382 | 1382 | 1375 | 1358 | 1340 | 1330 | 1330 | 1330 | 1330 | 1330 |
| (15/15/15) | GA-LS | 1114 | 1102 | 1035 | 1028 | 1017 | 1017 | 991 | 988 | 988 | 988 | 988 | 979 | 979 | 973 | 973 | 973 | 970 | | | |
| | GA-Mce1 | 1131 | 1119 | 1022 | 1001 | 987 | 981 | 968 | 968 | 967 | 967 | 964 | 964 | 964 | 964 | 964 | 956 | 954 | | | |
| | GA-Mce4 | 1131 | 1119 | 1018 | 1012 | 1012 | 1000 | 990 | 990 | 989 | 980 | 978 | 978 | 971 | 965 | 953 | 950 | 950 | | | |

TABLEAU 2.9: Convergence des algorithmes pour une simulation
pour les instances de (Adams et al., 1988) avec la règle vdata

| Convergence des algorithmes pour une simulation pour les instances d'Adams avec la règle vdata (suite) | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | |
| 101 | GA | 735 | 425 | 412 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 |
| (5/5/10) | GA-LS | 425 | 412 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | |
| | GA-Mce1 | 425 | 412 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | | |
| | GA-Mce4 | 425 | 412 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 411 | 408 | 408 | 403 | | | |
| 102 | GA | 782 | 478 | 475 | 448 | 447 | 447 | 441 | 432 | 431 | 430 | 430 | 430 | 430 | 428 | 427 | 427 | 427 | 427 | 427 | 427 |
| (5/5/10) | GA-LS | 501 | 476 | 476 | 459 | 445 | 445 | 436 | 432 | 428 | 426 | 426 | 426 | 426 | 426 | 426 | 426 | 426 | 426 | 426 | |
| | GA-Mce1 | 501 | 476 | 476 | 461 | 459 | 448 | 448 | 440 | 434 | 424 | 424 | 424 | 424 | 424 | 424 | 424 | 424 | 424 | | |
| | GA-Mce4 | 478 | 472 | 457 | 452 | 446 | 445 | 445 | 445 | 445 | 445 | 436 | 420 | 420 | 420 | 420 | 420 | 420 | | | |
| 103 | GA | 741 | 502 | 486 | 483 | 483 | 483 | 483 | 483 | 483 | 483 | 483 | 483 | 483 | 483 | 483 | 483 | 483 | 483 | 483 | 483 |
| (5/5/10) | GA-LS | 502 | 486 | 483 | 483 | 483 | 477 | 469 | 456 | 451 | 451 | 451 | 451 | 449 | 449 | 446 | 446 | 446 | 446 | 446 | |
| | GA-Mce1 | 510 | 487 | 476 | 463 | 442 | 438 | 438 | 438 | 436 | 436 | 436 | 436 | 436 | 436 | 436 | 436 | 436 | 427 | | |
| | GA-Mce4 | 510 | 487 | 476 | 463 | 442 | 438 | 436 | 436 | 436 | 428 | 428 | 427 | 427 | 413 | 413 | 405 | | | | |
| 104 | GA | 742 | 477 | 460 | 460 | 460 | 454 | 429 | 426 | 418 | 418 | 418 | 418 | 418 | 418 | 418 | 414 | 414 | 414 | 414 | 414 |
| (5/5/10) | GA-LS | 477 | 460 | 460 | 454 | 441 | 438 | 438 | 438 | 438 | 428 | 413 | 413 | 413 | 413 | 413 | 413 | 413 | 413 | 413 | |
| | GA-Mce1 | 477 | 460 | 454 | 441 | 432 | 426 | 415 | 415 | 413 | 413 | 413 | 413 | 413 | 413 | 413 | 413 | 413 | 413 | | |
| | GA-Mce4 | 477 | 460 | 454 | 454 | 444 | 443 | 443 | 437 | 427 | 424 | 421 | 421 | 421 | 421 | 421 | 419 | 413 | | | |
| 105 | GA | 714 | 422 | 413 | 401 | 385 | 380 | 380 | 380 | 380 | 380 | 380 | 380 | 380 | 380 | 380 | 380 | 380 | 380 | 380 | 379 |
| (5/5/10) | GA-LS | 463 | 437 | 425 | 412 | 412 | 406 | 388 | 388 | 381 | 381 | 378 | 373 | 373 | 373 | 373 | 373 | 373 | 373 | 373 | |
| | GA-Mce1 | 463 | 437 | 425 | 412 | 412 | 404 | 382 | 382 | 382 | 382 | 379 | 373 | 373 | 373 | 373 | 373 | 373 | 373 | | |
| | GA-Mce4 | 463 | 437 | 425 | 412 | 412 | 410 | 395 | 387 | 383 | 380 | 376 | 376 | 373 | 373 | 373 | 373 | 373 | | | |
| 106 | GA | 1085 | 727 | 680 | 676 | 676 | 676 | 671 | 667 | 667 | 667 | 667 | 667 | 667 | 667 | 667 | 667 | 667 | 667 | 667 | 667 |

| Convergence des algorithmes pour une simulation pour les instances d'Adams (suite) | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | |
| (5/5/15) | GA-LS | 727 | 680 | 676 | 676 | 669 | 667 | 667 | 667 | 667 | 667 | 667 | 667 | 667 | 667 | 667 | 667 | 667 | 667 | | |
| | GA-Mce1 | 727 | 680 | 676 | 676 | 676 | 676 | 676 | 667 | 667 | 667 | 667 | 667 | 667 | 667 | 667 | 646 | 646 | | | |
| | GA-Mce4 | 727 | 680 | 676 | 676 | 676 | 676 | 667 | 667 | 667 | 667 | 667 | 667 | 667 | 667 | 667 | 667 | 667 | | | |
| 107 | GA | 1054 | 668 | 668 | 660 | 654 | 640 | 640 | 640 | 640 | 637 | 637 | 632 | 631 | 626 | 626 | 626 | 619 | 619 | 617 | 617 |
| (5/4/15) | GA-LS | 741 | 690 | 676 | 656 | 640 | 640 | 623 | 617 | 617 | 617 | 616 | 616 | 616 | 612 | 612 | 612 | 612 | 604 | | |
| | GA-Mce1 | 741 | 690 | 676 | 656 | 640 | 640 | 622 | 616 | 616 | 616 | 601 | 601 | 601 | 601 | 600 | 600 | 600 | | | |
| | GA-Mce4 | 741 | 690 | 676 | 656 | 640 | 640 | 622 | 616 | 616 | 616 | 616 | 616 | 616 | 614 | 611 | 604 | 604 | | | |
| 108 | GA | 997 | 736 | 727 | 706 | 667 | 667 | 667 | 662 | 662 | 662 | 662 | 662 | 662 | 662 | 662 | 662 | 660 | 660 | 660 | 659 |
| (5/4/15) | GA-LS | 671 | 658 | 647 | 647 | 641 | 640 | 632 | 625 | 619 | 619 | 619 | 619 | 619 | 619 | 619 | 619 | 608 | 608 | 605 | |
| | GA-Mce1 | 671 | 658 | 647 | 647 | 647 | 641 | 640 | 627 | 610 | 605 | 605 | 605 | 604 | 604 | 604 | 604 | 604 | | | |
| | GA-Mce4 | 671 | 658 | 647 | 647 | 647 | 640 | 640 | 622 | 620 | 609 | 609 | 609 | 597 | 597 | 597 | 597 | 597 | | | |
| 109 | GA | 1072 | 697 | 693 | 656 | 656 | 656 | 656 | 656 | 656 | 656 | 656 | 656 | 656 | 656 | 656 | 656 | 656 | 656 | 656 | 656 |
| (5/5/15) | GA-LS | 722 | 704 | 701 | 689 | 682 | 676 | 654 | 646 | 646 | 646 | 646 | 646 | 646 | 646 | 646 | 646 | 646 | 646 | | |
| | GA-Mce1 | 722 | 704 | 698 | 688 | 688 | 667 | 660 | 649 | 630 | 630 | 630 | 630 | 630 | 630 | 630 | 630 | 630 | | | |
| | GA-Mce4 | 722 | 704 | 698 | 688 | 682 | 676 | 676 | 669 | 667 | 667 | 660 | 660 | 634 | 634 | 631 | 629 | 621 | | | |
| 110 | GA | 989 | 694 | 683 | 661 | 660 | 655 | 655 | 655 | 643 | 637 | 637 | 627 | 625 | 625 | 625 | 623 | 623 | 623 | 623 | 623 |
| (5/5/15) | GA-LS | 662 | 624 | 620 | 610 | 610 | 609 | 609 | 609 | 604 | 604 | 604 | 604 | 604 | 604 | 599 | 599 | 599 | 599 | 599 | |
| | GA-Mce1 | 662 | 624 | 620 | 609 | 609 | 599 | 599 | 599 | 599 | 599 | 599 | 599 | 599 | 599 | 599 | 599 | 599 | | | |
| | GA-Mce4 | 662 | 624 | 620 | 609 | 609 | 607 | 599 | 599 | 599 | 597 | 597 | 597 | 597 | 597 | 597 | 597 | | | | |
| 111 | GA | 1391 | 976 | 931 | 925 | 910 | 910 | 910 | 901 | 901 | 901 | 901 | 901 | 901 | 901 | 901 | 901 | 871 | 845 | 836 | 836 |
| (5/5/20) | GA-LS | 935 | 923 | 889 | 860 | 845 | 838 | 827 | 827 | 827 | 824 | 824 | 824 | 824 | 824 | 824 | 824 | 824 | 824 | 824 | |
| | GA-Mce1 | 935 | 923 | 889 | 860 | 845 | 838 | 835 | 835 | 824 | 824 | 824 | 824 | 824 | 824 | 824 | 824 | 824 | | | |
| | GA-Mce4 | 935 | 923 | 889 | 860 | 845 | 838 | 835 | 835 | 832 | 829 | 829 | 829 | 824 | 824 | 824 | | | | | |
| 112 | GA | 1343 | 987 | 943 | 922 | 893 | 865 | 865 | 865 | 862 | 862 | 859 | 859 | 859 | 859 | 856 | 852 | 852 | 849 | 849 | 849 |

| Convergence des algorithmes pour une simulation pour les instances d'Adams (suite) | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | |
| (5/5/20) | GA-LS | 987 | 943 | 922 | 893 | 865 | 865 | 865 | 865 | 865 | 865 | 865 | 862 | 862 | 862 | 862 | 862 | 862 | 861 | 847 | |
| | GA-Mce1 | 987 | 943 | 922 | 893 | 865 | 865 | 864 | 861 | 859 | 859 | 859 | 859 | 859 | 859 | 841 | 832 | 832 | | | |
| | GA-Mce4 | 987 | 943 | 922 | 893 | 865 | 865 | 854 | 840 | 838 | 837 | 833 | 833 | 833 | 831 | 826 | 826 | 826 | | | |
| 113 | GA | 1469 | 1005 | 971 | 951 | 930 | 923 | 922 | 922 | 922 | 922 | 908 | 892 | 892 | 881 | 874 | 868 | 864 | 864 | 864 | 864 |
| (5/5/20) | GA-LS | 1005 | 971 | 953 | 932 | 922 | 907 | 900 | 900 | 887 | 884 | 883 | 883 | 875 | 875 | 875 | 873 | 871 | 870 | 869 | |
| | GA-Mce1 | 1005 | 971 | 953 | 932 | 922 | 904 | 895 | 895 | 893 | 883 | 867 | 866 | 866 | 866 | 866 | 866 | 866 | | | |
| | GA-Mce4 | 1005 | 971 | 953 | 932 | 922 | 914 | 895 | 890 | 875 | 875 | 875 | 869 | 861 | 855 | 855 | 855 | | | | |
| 114 | GA | 1458 | 1005 | 1004 | 1004 | 969 | 960 | 959 | 928 | 928 | 928 | 928 | 928 | 928 | 928 | 917 | 912 | 912 | 912 | 906 | 906 |
| (5/5/20) | GA-LS | 1030 | 1010 | 986 | 975 | 975 | 975 | 954 | 919 | 919 | 917 | 917 | 902 | 900 | 900 | 900 | 894 | 894 | 894 | | |
| | GA-Mce1 | 1030 | 1010 | 986 | 975 | 975 | 960 | 919 | 911 | 897 | 897 | 897 | 897 | 893 | 893 | 893 | 893 | 893 | | | |
| | GA-Mce4 | 1030 | 1010 | 986 | 975 | 975 | 970 | 920 | 918 | 917 | 914 | 913 | 913 | 911 | 911 | 901 | 894 | 894 | | | |
| 115 | GA | 1374 | 1042 | 967 | 948 | 922 | 910 | 906 | 902 | 902 | 902 | 902 | 895 | 888 | 887 | 887 | 887 | 887 | 879 | 879 | 879 |
| (5/5/20) | GA-LS | 932 | 892 | 882 | 875 | 869 | 862 | 862 | 862 | 862 | 862 | 862 | 862 | 862 | 859 | 859 | 859 | 859 | 859 | | |
| | GA-Mce1 | 932 | 892 | 882 | 875 | 870 | 870 | 870 | 864 | 862 | 861 | 859 | 854 | 854 | 850 | 850 | 850 | 850 | | | |
| | GA-Mce4 | 932 | 892 | 882 | 875 | 875 | 874 | 874 | 874 | 874 | 866 | 866 | 866 | 866 | 862 | 854 | 852 | 846 | | | |
| 116 | GA | 1196 | 692 | 644 | 633 | 630 | 617 | 613 | 613 | 613 | 613 | 613 | 606 | 592 | 561 | 561 | 561 | 555 | 545 | 543 | 543 |
| (10/10/10) | GA-LS | 692 | 644 | 633 | 606 | 599 | 565 | 559 | 550 | 547 | 540 | 538 | 538 | 536 | 536 | 536 | 536 | 536 | 536 | | |
| | GA-Mce1 | 692 | 644 | 633 | 606 | 552 | 545 | 538 | 535 | 535 | 530 | 530 | 526 | 526 | 510 | 510 | 510 | 510 | | | |
| | GA-Mce4 | 692 | 644 | 633 | 606 | 563 | 528 | 520 | 520 | 520 | 518 | 515 | 515 | 515 | 504 | 504 | 501 | | | | |
| 117 | GA | 1185 | 745 | 684 | 644 | 639 | 622 | 620 | 620 | 619 | 619 | 612 | 611 | 611 | 611 | 611 | 606 | 606 | 601 | 586 | 584 |
| (10/10/10) | GA-LS | 728 | 685 | 646 | 634 | 634 | 632 | 632 | 632 | 596 | 582 | 582 | 580 | 578 | 578 | 572 | 572 | 572 | 572 | 572 | |
| | GA-Mce1 | 728 | 685 | 646 | 634 | 634 | 628 | 622 | 622 | 622 | 622 | 603 | 597 | 585 | 585 | 579 | 562 | 561 | 554 | | |
| | GA-Mce4 | 728 | 685 | 646 | 634 | 634 | 632 | 632 | 632 | 613 | 600 | 590 | 584 | 582 | 572 | 550 | 541 | 541 | 541 | | |
| 118 | GA | 1157 | 737 | 721 | 674 | 645 | 637 | 635 | 626 | 626 | 618 | 618 | 618 | 616 | 610 | 604 | 604 | 599 | 594 | 574 | 559 |

| Convergence des algorithmes pour une simulation pour les instances d'Adams (suite) | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | |
| (10/10/10) | GA-LS | 406 | 406 | 395 | 395 | 395 | 395 | 395 | 395 | 390 | 390 | 390 | 390 | 390 | 390 | 390 | 390 | 390 | 390 | | |
| | GA-Mce1 | 406 | 396 | 395 | 395 | 395 | 395 | 390 | 390 | 390 | 390 | 390 | 390 | 390 | 390 | 390 | 390 | | | | |
| | GA-Mce4 | 406 | 396 | 395 | 395 | 395 | 395 | 387 | 387 | 387 | 387 | 387 | 387 | 382 | 382 | 382 | | | | | |
| 119 | GA | 1073 | 661 | 600 | 598 | 584 | 579 | 575 | 572 | 565 | 555 | 538 | 538 | 538 | 532 | 528 | 518 | 518 | 518 | 508 | 505 |
| (10/10/10) | GA-LS | 403 | 375 | 375 | 375 | 373 | 373 | 373 | 373 | 373 | 373 | 373 | 373 | 373 | 373 | 373 | 373 | 365 | 349 | | |
| | GA-Mce1 | 403 | 378 | 352 | 346 | 338 | 338 | 338 | 335 | 335 | 332 | 332 | 332 | 327 | 327 | 327 | 327 | | | | |
| | GA-Mce4 | 403 | 378 | 352 | 346 | 338 | 338 | 338 | 338 | 338 | 335 | 335 | 335 | 334 | 334 | | | | | | |
| 120 | GA | 1130 | 680 | 609 | 582 | 573 | 559 | 552 | 545 | 542 | 536 | 513 | 505 | 483 | 483 | 469 | 467 | 467 | 467 | 465 | 465 |
| (10/10/10) | GA-LS | 404 | 404 | 402 | 402 | 400 | 398 | 398 | 398 | 396 | 396 | 396 | 396 | 396 | 396 | 396 | 396 | 396 | | | |
| | GA-Mce1 | 404 | 404 | 398 | 396 | 391 | 384 | 379 | 374 | 374 | 370 | 370 | 366 | 355 | 353 | 353 | | | | | |
| | GA-Mce4 | 404 | 404 | 398 | 398 | 391 | 388 | 384 | 384 | 384 | 384 | 384 | 382 | 379 | 379 | 379 | | | | | |
| 121 | GA | 1719 | 1184 | 1168 | 1129 | 1110 | 1058 | 1007 | 1005 | 995 | 959 | 925 | 925 | 905 | 873 | 873 | 873 | 868 | 863 | 855 | 842 |
| (10/10/15) | GA-LS | 628 | 628 | 626 | 608 | 608 | 608 | 608 | 608 | 608 | 588 | 582 | 582 | 582 | 581 | 576 | 576 | 575 | 575 | | |
| | GA-Mce1 | 632 | 604 | 596 | 583 | 582 | 582 | 582 | 582 | 578 | 573 | 573 | 573 | 573 | 573 | 573 | 572 | | | | |
| | GA-Mce4 | 632 | 604 | 596 | 583 | 582 | 582 | 582 | 582 | 581 | 576 | 571 | 565 | 560 | 555 | 552 | | | | | |
| 122 | GA | 1597 | 1091 | 1013 | 961 | 924 | 896 | 878 | 862 | 861 | 843 | 839 | 833 | 830 | 818 | 818 | 817 | 817 | 811 | 810 | 798 |
| (10/10/15) | GA-LS | 594 | 559 | 552 | 551 | 548 | 543 | 537 | 530 | 530 | 530 | 530 | 530 | 525 | 524 | 521 | 521 | 521 | 519 | | |
| | GA-Mce1 | 594 | 574 | 557 | 544 | 540 | 535 | 535 | 533 | 533 | 527 | 527 | 527 | 527 | 524 | 520 | 519 | | | | |
| | GA-Mce4 | 594 | 574 | 557 | 544 | 540 | 536 | 534 | 534 | 532 | 532 | 532 | 532 | 529 | 526 | 525 | 516 | | | | |
| 123 | GA | 1604 | 1040 | 1005 | 971 | 938 | 891 | 887 | 876 | 871 | 863 | 845 | 823 | 820 | 815 | 814 | 814 | 810 | 809 | 798 | 795 |
| (10/10/15) | GA-LS | 505 | 504 | 502 | 502 | 502 | 499 | 492 | 486 | 481 | 480 | 480 | 480 | 477 | 477 | 477 | 474 | 474 | 474 | 474 | |
| | GA-Mce1 | 505 | 504 | 499 | 494 | 494 | 494 | 492 | 492 | 491 | 491 | 485 | 483 | 482 | 479 | 479 | 479 | 479 | 478 | | |
| | GA-Mce4 | 505 | 504 | 494 | 489 | 484 | 484 | 484 | 479 | 479 | 476 | 474 | 474 | 474 | 460 | 460 | 459 | | | | |
| 124 | GA | 1443 | 1040 | 970 | 940 | 933 | 920 | 908 | 902 | 902 | 891 | 866 | 854 | 850 | 840 | 813 | 808 | 808 | 808 | 808 | 798 |

| Convergence des algorithmes pour une simulation pour les instances d'Adams (suite) | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | |
| (10/10/15) | GA-LS | 574 | 552 | 534 | 531 | 515 | 501 | 501 | 498 | 498 | 492 | 481 | 480 | 474 | 474 | 474 | 473 | 473 | 473 | 469 | 467 |
| | GA-Mce1 | 574 | 560 | 529 | 500 | 497 | 466 | 464 | 464 | 464 | 464 | 464 | 463 | 463 | 462 | 461 | 452 | 452 | 452 | 452 | |
| | GA-Mce4 | 574 | 560 | 529 | 500 | 497 | 479 | 478 | 477 | 477 | 477 | 463 | 460 | 453 | 446 | 441 | | | | | |
| 125 | GA | 1412 | 979 | 963 | 924 | 879 | 865 | 865 | 864 | 863 | 862 | 851 | 845 | 834 | 813 | 803 | 801 | 771 | 771 | 766 | 757 |
| (10/10/15) | GA-LS | 593 | 531 | 511 | 511 | 508 | 508 | 507 | 481 | 474 | 474 | 474 | 474 | 474 | 467 | 467 | 467 | 467 | 467 | 467 | |
| | GA-Mce1 | 607 | 559 | 540 | 512 | 496 | 493 | 482 | 479 | 468 | 467 | 467 | 467 | 467 | 467 | 467 | 465 | 465 | 465 | | |
| | GA-Mce4 | 607 | 559 | 540 | 512 | 496 | 493 | 482 | 479 | 476 | 476 | 466 | 466 | 466 | 466 | 466 | | | | | |
| 126 | GA | 2009 | 1458 | 1370 | 1289 | 1267 | 1249 | 1244 | 1221 | 1213 | 1213 | 1213 | 1213 | 1213 | 1205 | 1204 | 1200 | 1180 | 1161 | 1161 | 1158 |
| (10/10/20) | GA-LS | 796 | 761 | 761 | 728 | 710 | 686 | 686 | 680 | 676 | 676 | 676 | 676 | 676 | 676 | 676 | 676 | 676 | 676 | 676 | |
| | GA-Mce1 | 796 | 738 | 723 | 723 | 715 | 706 | 699 | 690 | 689 | 689 | 677 | 677 | 677 | 676 | 670 | 670 | 670 | 670 | | |
| | GA-Mce4 | 796 | 738 | 723 | 723 | 723 | 704 | 693 | 685 | 673 | 673 | 673 | 673 | 671 | 662 | 662 | 662 | 662 | | | |
| 127 | GA | 1978 | 1506 | 1408 | 1365 | 1298 | 1268 | 1228 | 1218 | 1197 | 1173 | 1163 | 1140 | 1127 | 1127 | 1127 | 1123 | 1111 | 1111 | 1110 | 1110 |
| (10/10/20) | GA-LS | 759 | 738 | 715 | 669 | 663 | 661 | 653 | 653 | 653 | 653 | 649 | 648 | 648 | 648 | 648 | 648 | 648 | 648 | 648 | |
| | GA-Mce1 | 759 | 747 | 697 | 693 | 671 | 669 | 658 | 653 | 648 | 648 | 648 | 642 | 642 | 642 | 642 | 642 | 639 | 635 | | |
| | GA-Mce4 | 759 | 747 | 700 | 695 | 683 | 682 | 682 | 682 | 682 | 673 | 671 | 654 | 654 | 649 | 644 | 644 | 644 | 639 | | |
| 128 | GA | 1865 | 1366 | 1336 | 1303 | 1270 | 1240 | 1146 | 1140 | 1139 | 1129 | 1128 | 1121 | 1118 | 1118 | 1118 | 1118 | 1118 | 1118 | 1118 | 1115 |
| (10/10/20) | GA-LS | 679 | 653 | 653 | 649 | 643 | 638 | 636 | 631 | 628 | 621 | 620 | 615 | 615 | 615 | 615 | 615 | 613 | 601 | 601 | |
| | GA-Mce1 | 680 | 672 | 652 | 646 | 630 | 623 | 623 | 623 | 623 | 623 | 619 | 618 | 614 | 610 | 603 | 603 | 602 | 596 | | |
| | GA-Mce4 | 680 | 672 | 652 | 646 | 635 | 628 | 628 | 628 | 623 | 621 | 621 | 611 | 604 | 604 | 597 | 591 | 591 | 586 | | |
| 129 | GA | 2022 | 1336 | 1268 | 1213 | 1173 | 1151 | 1135 | 1089 | 1062 | 1062 | 1059 | 1053 | 1030 | 1001 | 1001 | 988 | 969 | 969 | 969 | 969 |
| (10/10/20) | GA-LS | 640 | 629 | 607 | 607 | 602 | 600 | 597 | 594 | 594 | 594 | 594 | 594 | 594 | 594 | 592 | 592 | 592 | 592 | 592 | |
| | GA-Mce1 | 640 | 625 | 617 | 617 | 614 | 613 | 605 | 598 | 591 | 591 | 588 | 588 | 588 | 588 | 583 | 577 | 577 | 577 | | |
| | GA-Mce4 | 640 | 625 | 617 | 617 | 603 | 602 | 602 | 596 | 581 | 581 | 581 | 581 | 580 | 580 | 580 | 574 | 574 | | | |
| 130 | GA | 2081 | 1353 | 1266 | 1228 | 1217 | 1211 | 1158 | 1128 | 1128 | 1121 | 1114 | 1063 | 1044 | 1029 | 1029 | 1023 | 988 | 969 | 956 | 949 |

| Convergence des algorithmes pour une simulation pour les instances d'Adams (suite) | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | |
| (10/10/20) | GA-LS | 706 | 664 | 621 | 608 | 608 | 608 | 608 | 608 | 608 | 608 | 608 | 608 | 598 | 595 | 582 | 581 | 581 | 581 | 581 | |
| | GA-Mce1 | 706 | 673 | 621 | 615 | 604 | 591 | 589 | 589 | 589 | 578 | 578 | 578 | 577 | 577 | 572 | 569 | 566 | 566 | | |
| | GA-Mce4 | 706 | 673 | 621 | 615 | 615 | 613 | 608 | 606 | 595 | 593 | 593 | 592 | 589 | 581 | 581 | 574 | 574 | | | |
| 131 | GA | 2882 | 2189 | 2043 | 1951 | 1892 | 1876 | 1839 | 1824 | 1807 | 1772 | 1764 | 1745 | 1693 | 1693 | 1675 | 1660 | 1660 | 1636 | 1614 | 1614 |
| (10/10/30) | GA-LS | 1081 | 1041 | 981 | 964 | 964 | 964 | 961 | 945 | 924 | 915 | 914 | 912 | 905 | 905 | 904 | 896 | 895 | 895 | 895 | |
| | GA-Mce1 | 951 | 903 | 871 | 861 | 851 | 851 | 847 | 825 | 819 | 819 | 819 | 818 | 817 | 817 | 817 | 817 | 817 | | | |
| | GA-Mce4 | 951 | 903 | 871 | 861 | 851 | 851 | 851 | 841 | 813 | 797 | 795 | 795 | 789 | 789 | 779 | | | | | |
| 132 | GA | 2823 | 2014 | 1907 | 1899 | 1864 | 1828 | 1826 | 1819 | 1772 | 1754 | 1750 | 1745 | 1736 | 1715 | 1698 | 1692 | 1692 | 1692 | 1685 | 1684 |
| (10/10/30) | GA-LS | 1063 | 1056 | 1014 | 982 | 978 | 973 | 969 | 966 | 965 | 965 | 960 | 960 | 959 | 955 | 953 | 948 | 939 | 939 | | |
| | GA-Mce1 | 1066 | 1054 | 1037 | 1013 | 1004 | 987 | 983 | 967 | 958 | 953 | 941 | 941 | 932 | 928 | 928 | 920 | 920 | | | |
| | GA-Mce4 | 1066 | 1054 | 1037 | 1013 | 1004 | 964 | 940 | 931 | 924 | 924 | 918 | 917 | 917 | 917 | 912 | 905 | | | | |
| 133 | GA | 2812 | 2295 | 2042 | 1949 | 1928 | 1918 | 1881 | 1862 | 1860 | 1849 | 1846 | 1845 | 1829 | 1780 | 1756 | 1756 | 1752 | 1752 | 1748 | 1748 |
| (10/10/30) | GA-LS | 1012 | 985 | 981 | 976 | 964 | 957 | 957 | 952 | 948 | 944 | 939 | 939 | 933 | 932 | 924 | 924 | 924 | 924 | 924 | |
| | GA-Mce1 | 1040 | 1016 | 982 | 969 | 958 | 926 | 925 | 925 | 925 | 917 | 915 | 915 | 910 | 910 | 910 | 899 | 899 | | | |
| | GA-Mce4 | 1040 | 1016 | 982 | 969 | 958 | 952 | 930 | 923 | 920 | 920 | 920 | 910 | 904 | 904 | 904 | 897 | | | | |
| 134 | GA | 2673 | 1982 | 1897 | 1809 | 1778 | 1764 | 1734 | 1685 | 1671 | 1626 | 1600 | 1600 | 1600 | 1591 | 1586 | 1580 | 1560 | 1530 | 1524 | 1512 |
| (10/10/30) | GA-LS | 883 | 860 | 832 | 831 | 829 | 829 | 822 | 822 | 821 | 821 | 814 | 814 | 814 | 805 | 805 | 805 | 805 | 805 | 805 | |
| | GA-Mce1 | 883 | 883 | 874 | 859 | 848 | 838 | 834 | 825 | 825 | 823 | 814 | 814 | 813 | 813 | 813 | 810 | 805 | 795 | | |
| | GA-Mce4 | 883 | 883 | 883 | 883 | 883 | 858 | 848 | 848 | 831 | 809 | 803 | 801 | 798 | 790 | 787 | 785 | 785 | | | |
| 135 | GA | 2887 | 2047 | 1976 | 1950 | 1935 | 1885 | 1862 | 1833 | 1817 | 1764 | 1742 | 1704 | 1673 | 1633 | 1626 | 1626 | 1623 | 1576 | 1576 | 1576 |
| (10/10/30) | GA-LS | 950 | 937 | 924 | 905 | 871 | 865 | 857 | 854 | 849 | 830 | 830 | 827 | 822 | 817 | 816 | 816 | 816 | 816 | | |
| | GA-Mce1 | 951 | 903 | 871 | 861 | 851 | 851 | 847 | 825 | 819 | 819 | 819 | 818 | 817 | 817 | 817 | 817 | 817 | | | |
| | GA-Mce4 | 951 | 903 | 871 | 861 | 851 | 851 | 851 | 841 | 813 | 797 | 795 | 795 | 789 | 789 | 779 | | | | | |
| 136 | GA | 554 | 505 | 481 | 481 | 481 | 479 | 479 | 479 | 476 | 473 | 462 | 459 | 459 | 459 | 459 | 459 | 457 | 457 | 456 | 456 |

| Convergence des algorithmes pour une simulation pour les instances d'Adams (suite) | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Algorithme | Convergence | | | | | | | | | | | | | | | | | | | |
| (15/15/15) | GA-LS | 503 | 484 | 484 | 475 | 475 | 475 | 471 | 471 | 471 | 468 | 465 | 465 | 465 | 461 | 461 | 461 | 461 | 461 | 451 | |
| | GA-Mce1 | 504 | 497 | 497 | 497 | 497 | 471 | 466 | 459 | 450 | 450 | 443 | 443 | 443 | 443 | 441 | 441 | 440 | 440 | | |
| | GA-Mce4 | 504 | 460 | 451 | 451 | 451 | 451 | 451 | 449 | 449 | 449 | 449 | 449 | 449 | 449 | 449 | 449 | 449 | 447 | | |
| 137 | GA | 626 | 546 | 531 | 524 | 521 | 521 | 521 | 521 | 501 | 497 | 497 | 493 | 491 | 489 | 489 | 489 | 489 | 489 | 489 | 489 |
| (15/15/15) | GA-LS | 546 | 532 | 524 | 520 | 520 | 520 | 513 | 496 | 496 | 493 | 492 | 492 | 492 | 492 | 491 | 491 | 491 | 491 | 491 | |
| | GA-Mce1 | 546 | 531 | 524 | 514 | 514 | 514 | 514 | 499 | 491 | 490 | 486 | 485 | 485 | 485 | 485 | 485 | 485 | 476 | | |
| | GA-Mce4 | 546 | 531 | 524 | 514 | 514 | 514 | 506 | 506 | 506 | 501 | 501 | 501 | 501 | 501 | 501 | 501 | 501 | 481 | | |
| 138 | GA | 783 | 734 | 722 | 708 | 668 | 663 | 658 | 647 | 647 | 647 | 642 | 642 | 635 | 634 | 632 | 627 | 622 | 622 | 622 | 622 |
| (15/15/15) | GA-LS | 658 | 656 | 643 | 643 | 643 | 643 | 636 | 636 | 636 | 636 | 633 | 633 | 630 | 630 | 625 | 624 | 624 | 618 | 609 | |
| | GA-Mce1 | 658 | 656 | 637 | 636 | 636 | 630 | 630 | 627 | 626 | 625 | 625 | 625 | 622 | 615 | 609 | 609 | 603 | 603 | | |
| | GA-Mce4 | 658 | 656 | 648 | 643 | 642 | 642 | 642 | 638 | 637 | 636 | 630 | 624 | 617 | 607 | 607 | 607 | 607 | 607 | | |
| 139 | GA | 684 | 612 | 589 | 585 | 580 | 580 | 580 | 580 | 580 | 580 | 575 | 575 | 571 | 570 | 570 | 568 | 568 | 568 | 568 | 568 |
| (15/15/15) | GA-LS | 615 | 590 | 565 | 565 | 561 | 558 | 557 | 555 | 555 | 555 | 555 | 553 | 551 | 551 | 551 | 551 | 551 | 551 | 551 | |
| | GA-Mce1 | 617 | 614 | 614 | 609 | 607 | 604 | 595 | 560 | 552 | 548 | 548 | 548 | 548 | 548 | 545 | 540 | 539 | 536 | | |
| | GA-Mce4 | 617 | 600 | 576 | 570 | 570 | 563 | 546 | 544 | 543 | 542 | 535 | 535 | 535 | 533 | 531 | 528 | 527 | 520 | | |
| 140 | GA | 653 | 552 | 550 | 545 | 542 | 540 | 539 | 539 | 539 | 539 | 539 | 528 | 520 | 520 | 520 | 516 | 504 | 504 | 504 | 495 |
| (15/15/15) | GA-LS | 547 | 545 | 532 | 528 | 489 | 481 | 481 | 480 | 480 | 480 | 480 | 480 | 480 | 480 | 480 | 480 | 477 | 477 | 472 | |
| | GA-Mce1 | 552 | 550 | 550 | 545 | 545 | 545 | 545 | 545 | 545 | 545 | 545 | 545 | 528 | 507 | 481 | 479 | 479 | 479 | | |
| | GA-Mce4 | 552 | 550 | 533 | 525 | 499 | 489 | 488 | 480 | 480 | 480 | 478 | 476 | 474 | 474 | 474 | 474 | 469 | 469 | | |

L'analyse du tableau 2.6 montre que les hybridations GA-(LS, Mce1 et Mce4) améliorent la convergence et les résultats par rapport au GA classique sur les instances de Brandimate. Sur la plupart des instances, les hybridations GA-(Mce1 et Mce4) améliorent la convergence de la recherche locale, à l'exception sur les instances mk3, mk4 et mk5. Pour les trois hybridations, la note finale est fréquemment équivalente voire même identique. C'est le cas généralement sur de petites instances tel que mk1, mk2, mk3, mk4, mk5, mk7 et mk13. Sur les instances mk6, mk8, mk9, mk10, mk11 et mk12 les hybridations GA-(Mce1 et Mce4) obtiennent de meilleurs résultats que la recherche locale. L'avantage des hybridations GA-(Mce1 et Mce4) est l'apport d'un supplément d'intensification et de diversification dans la recherche de solutions. Le GA-Mce4 diversifie davantage que le GA-Mce1 ce qui lui permet d'avoir les meilleurs résultats sur les instances mk1, mk2, mk8, mk11, mk13 et mk14. Par contre, cette diversification peut égarer la métaheuristique, par exemple sur les instances mk3, mk4 et mk5.

Logiquement pour les instances de Dauzérès-Pérès (voir tableau 2.7) les hybridations surpassent le GA original. Les hybridations GA-(Mce1 et Mce4) améliorent les résultats de la recherche locale, à l'exception des instances 5a 10a, 12a et 13a où ils sont égaux. Pour la convergence, le GA-Mce1 est plus rapide sur 12 instances, le GA-Mce4 est plus rapide sur 4 instances, par contre le GA-LS reste le plus rapide sur les instances 4a et 8a. Le GA-Mce4 est plus lent pour la convergence que GA-Mce1 mais elle obtient de meilleurs résultats sur 8 instances. Le GA-Mce1 améliore les performances de le GA-LS en appliquant une recherche locale sur les solutions dans la zone de la meilleure solution. Le GA-Mce4 est plus lent à la convergence car il diversifie la recherche mais il obtient fréquemment de meilleurs résultats. Mais le GA-Mce4 fait une contre-performance par rapport aux GA-(LS et Mce1) sur les instances 5a, 10a, 12a et 13a.

Les instances d'Adams sont des instances du *Job Shop Problem* transformées par des règles rdata et vdata qui qualifient plusieurs machines pour une opération. Pour chaque règle, 40 instances sont générées. Le GA est dominé par les trois hybridations GA-(LS, Mce1 et Mce4) sur les 80 instances. Sur les instances générées par rdata (voir tableau 2.8), les hybridations GA-(Mce1 et Mce4) améliorent légèrement les résultats par rapport à la recherche locale. Le GA-Mce4 obtient de meilleurs résultats dans 26 instances et le GA-Mce1 sur les 14 autres instances. Pour la convergence, le GA-LS reste plus rapide sur 9 instances 115, 119, 122, 125, 126, 127, 130, 133 et 135. Par contre le GA-Mce1 est la plus rapide sur 18 instances et le GA-Mce4 sur 13 instances. La diversification du GA-Mce4 est généralement un avantage mais elle la plus lente sur 9 instances : 109, 111, 115, 122, 125, 127, 130, 133 et 135. Tandis que le GA-Mce1 n'est le plus lent que sur 5 instances : 106, 116, 118, 119 et 126. La recherche locale garde la meilleure convergence sur 9 instances.

La règle vdata qualifie plus de machines pour une opération que la règle rdata. Les hybridations GA-(Mce1 et Mce4) améliorent les résultats du GA-LS à l'exception des instances 114, 119, 129 et 137 (voir tableau 2.9). L'hybridation GA-Mce4 obtient les meilleurs résultats sur 23 instances et le GA-Mce1 sur 13 instances. Le GA-LS garde les meilleurs résultats sur les instances 114,119, 129 et 137. Toutefois les résultats obtenus par les trois hybridations restent proches, la différence se fait

sur la convergence. Le GA-Mce1 est le plus rapide sur 18 instances et le GA-Mce4 sur 13 instances. Par contre la recherche locale reste la plus rapide sur 9 instances.

Les hybridations effectuent moins d'itérations que le GA classique car les PRLs nécessitent des calculs supplémentaires. Les GA-(Mce1 et Mce4) améliorent les résultats et/ou la convergence de la recherche locale en ajoutant de l'intensification et de la diversification lors de l'exploration de l'espace de solutions. L'intensification de la recherche dans la zone de la meilleure solution du GA-Mce1 permet une accélération de la convergence de la recherche locale. La diversification dans les zones inexplorées permet de trouver de nouvelles solutions intéressantes pour l'exploration, mais parfois cela détériore les performances.

## 2.9 Conclusion

Ce chapitre présente une méthode pour représenter l'espace de solutions sur un axe unidimensionnel. Cet axe permet une localisation rapide et efficace d'une solution dans l'espace des variables. Par la collaboration de l'évaluation d'une solution sur l'axe des variables et celui des objectifs, une métrique évalue l'exploration de l'espace de solutions par une méthode d'optimisation. Avec cette évaluation, des zones sont identifiées, celles qui sont inexplorées et explorées. Plusieurs catégories sont distinguées pour les zones explorées : contenant les meilleures solutions ou faiblement/fortement explorée par la population. Pour améliorer la convergence et les performances d'une métaheuristique, des zones sont identifiées périodiquement. Dans les zones inexplorées des solutions sont créées et une recherche locale est appliquée sur toutes les solutions contenues dans ces zones.

Le problème NP-difficile du FJSP avec pour objectif la minimisation du *makespan* est utilisé pour tester cette nouvelle hybridation. Ce problème a l'avantage d'être très connu et bien défini, ce qui permet d'avoir de nombreuses références pour se comparer. Un algorithme de la littérature est implémenté et hybridé avec la MCE. Les hybridations testées sont la recherche locale classique, sur les solutions dans les zones *nd* et créées dans les *zi*. Les résultats montrent une amélioration de la convergence et des notes grâce à l'hybridation avec la MCE.

# Chapitre 3

# Adaptation de la Méthode de Conversion Espace de recherche pour l'optimisation multiobjectif

## 3.1 Introduction

Le chapitre précédent présente une procédure pour guider des méthodes d'optimisation pour un problème mono-objectif. Dans ce chapitre la MCE combinée avec le HC est adaptée aux problèmes multiobjectifs. Les principes de la MCE sont gardés, mais quelques changements sont nécessaires. Les métaheuristiques et le HC améliorent simultanément tous les objectifs. Le front de Pareto optimal est recherché par les métaheuristiques et le HC ne retient pour le voisinage que les solutions améliorant des objectifs sans détériorer l'évaluation des autres. Le FJSP est gardé comme problème de référence, ainsi que l'objectif de minimisation du *makespan*. L'objectif de produire les lots juste à temps est ajouté, pour rendre le problème multiobjectif. La production des lots juste à temps, est la somme des retards (en anglais Tardiness $T_i$) et des avances (ou Earliness $E_i$) sur les dates limites (ou *due date*s) par rapport aux dates de réalisation des lots dans l'ordonnancement. Après l'évaluation des solutions par la fonction objectif, la classification par front de Pareto est utilisée pour identifier les solutions non-dominées et par conséquent les zones contenants les meilleures solutions. Pour le test de convergence $Ind - MCE$, la métrique C (Ziztler et al., 2003) qui compare l'ancien et le nouveau front de Pareto, remplace la comparaison entre l'ancienne et la nouvelle note. Plusieurs métaheuristiques (NSGAII, SPEA2, MOAIS, MOACO, MOPSO et MOABC) sont comparées avec et sans l'hybridation de MCE avec le HC. Les résultats montrent que la MCE avec la HC améliore nettement les résultats des métaheuristiques.

## 3.2 État de l'art sur les métaheuristiques pour résoudre le FJSP multiobjectif

Cette section présente quelques méthodes d'optimisation approchées proposées récemment pour résoudre le FSJP MultiObjectif (MOFJSP). Le *Non-dominated Sorting Genetic Algorithm-II* (NSGAII) et le *Non-dominated Ranking Genetic Algorithm* (NRGA) (Rahmati et al., 2013) sont deux algorithmes évolutionnaires adaptés pour le FJSP en considérant les objectifs de minimisation du *makespan*, de la charge de travail sur les machines (en anglais *total work load of machines*) et de la charge sur la machine critique (en anglais *critical machine work load*). La représentation de la solution est équivalente à celle du mono-objectif (Tay and Wibowo, 2004) avec deux parties l'une pour le séquençage des opérations et l'autre pour l'affectation des opérations aux machines. Les opérateurs de croisement utilisés sont l'*Improved Precedence Operation crossover* (IPOX) et *Multipoint Preservative crossover* (MPX). L'IPOX s'applique à la partie séquençage des opérations, il copie dans l'enfant une partie des gènes du premier parent à la même place et complète en respectant l'ordre des gènes du deuxième parent. Le MPX est utilisé pour croiser la partie affectation des opérations aux machines. Un vecteur binaire de la taille de la partie affectation est généré selon une loi de probabilité uniforme. Lorsque le bit est à 0, le gène de l'enfant 1 appartient au premier parent et s'il est à 1 le gène viendra du deuxième parent, vice-versa pour l'obtention de l'enfant 2. Pour la partie séquencement, le déplacement d'un gène sert d'opérateur de mutation. La valeur d'un gène est modifiée pour l'opérateur de mutation dans la partie affectation. Les résultats montrent qu'ils dominent tous les deux un MOGA classique adapté pour le FJSP multiobjectif.

Un autre MOEA est proposé pour résoudre le FJSP avec les mêmes objectifs : minimisation du *makespan*, de la charge de travail sur les machines et sur la machine critique (Chiang and Lin, 2013). Les opérateurs de croisement sont le *Precedence Preserving Order-based Crossover* (POX) pour la partie ordonnancement et le *Assignment Crossover* (ASX) pour la partie affectation. Le POX sélectionne les opérations d'un lot sur le premier parent et les copie aux mêmes places dans l'enfant. Le reste des gènes est complété en respectant l'ordre des opérations dans le deuxième parent. L'ASX échange des gènes entre les parents pour constituer les enfants. Les opérateurs de mutation cherchent à améliorer un des objectifs en modifiant l'ordonnancement ou les affectations. Pour la partie ordonnancement, des opérations sont sélectionnées sur le chemin critique et échangent leurs places ou sont déplacées. Pour la partie affectation, des opérations sont réassignées pour diminuer la charge de travail ou réduire la *makespan*. Les résultats montrent que le nombre de solutions non-dominées est augmenté de 70% par rapport aux algorithmes comparés.

L'AG est aussi adapté pour le FJSP multiobjectif (Lan et al., 2010) en considérant les objectifs de minimisation du *makespan* et des coûts. L'originalité de cet algorithme repose sur l'utilisation des principes de l'AIS et du SA pour construire la population initiale et choisir les meilleurs paramètres pour les opérateurs de croisement et de mutation. Les expérimentations montrent que ce GA amélioré obtient de meilleurs résultats que le GA classique.

Pour assurer un bon départ pour un EA, il est important d'avoir une population initiale avec des solutions de qualité et diversifiées. Pour le FJSP, des heuristiques (Kacem et al., 2002a) sont utilisées pour fournir des solutions de bonne qualité et uniformément réparties dans l'espace des variables. Généralement, chaque heuristique est conçue pour l'un des sous-problèmes du FJSP, l'ordonnancement ou l'affectation des opérations. Ces heuristiques optimisent l'un des objectifs et pour cela ne sont utilisées que pour générer une partie des solutions de la population initiale. Pour assurer la diversification, des solutions sont générées aléatoirement. Grâce à cette population initiale de qualité, les EAs améliorent leurs résultats.

Le FSJP multiobjectif est aussi résolu par d'autres métaheuristiques tel que l'AIS (Lu, 2009). Le *Stretching Technique Based Clonal Selection Algorithm* (STCSA) résout un FJSP avec les objectifs de minimisation du *makespan*, de la charge de travail sur les machines et sur la machine critique (la plus chargée). Le STCSA utilise deux procédures la *Stretching Technique* (ST) et la *Clonal Selection Algorithm* (CSA). Le ST est une technique qui recherche un minimum global en utilisant des minimums locaux. La première étape trouve des minimums locaux et la seconde par exploration du voisinage des minimums locaux, recherche le minimum global. Le CSA est l'une des techniques de sélection les plus répandues pour les AIS (De Castro and Von Zuben, 2002).

Le *Double Layer ACO* (DLACO) (Xing et al., 2008) (Xing et al., 2009b) est une adaptation de l'ACO pour résoudre le FJSP avec les objectifs de minimiser le *makespan*, la somme de la charge de travail sur les machines et la charge sur la machine critique. La construction d'une solution se fait en deux étapes. Lors de la première étape, les opérations des lots sont affectées aux machines. L'ordre de passage des opérations est déterminé lors de la seconde étape. Il existe deux mémoires l'une pour l'affectation des opérations et l'autre pour leur ordonnancement. Une archive stocke les solutions non-dominées trouvées par l'algorithme. Les mémoires sont mises à jour avec les solutions dans le front de Pareto optimal obtenu à chaque itération. Pour les expérimentations, le DLACO est comparé à des MOEA, MOPSO et MOTS, il obtient des résultats équivalents ou meilleurs que ces algorithmes.

Initialement conçu pour l'optimisation dans le domaine continu, pour résoudre le FJSP le MOPSO s'adapte souvent avec une LS et une archive pour stocker les solutions non-dominées. Le *Discrete Particle Swarm Optimization* (DPSO) est une adaptation du MOPSO pour le problème combinatoire du FJSP (Shao et al., 2013). Les objectifs considérés sont la minimisation du *makespan*, de la charge de travail sur les machines et la machine critique. Le DPSO est une hybridation entre un PSO qui cherche l'optimum global et le SA qui est utilisé comme une LS pour trouver les optimums locaux. Le rang de Pareto et le *crowding distance* sont utilisés pour évaluer les particules. Le DPSO utilise une nouvelle stratégie où le déplacement des particules est influencé par toutes les solutions non-dominées. Une archive stocke toutes les solutions non-dominées, elle est mise à jour à chaque itération.

Une autre variante du MOPSO avec le SA (PSO+SA) est proposée (Xia and Wu, 2005) avec

les mêmes objectifs. Ce PSO+SA est composé de trois étapes, la première la population initiale est créée, la seconde un MOPSO fait une recherche dans l'espace de solutions et la dernière un SA améliore les solutions de la population.

L'hybridation du MOPSO avec le TS (PSO+TS) (Zhang et al., 2009) est aussi utilisé pour résoudre le FJSP avec les mêmes objectifs. Comme le PSO+SA, il est composé de trois étapes, l'initialisation, le MOPSO et le MOTS. La procédure est identique au PSO+SA, mais le SA est remplacé par le TS. Les expérimentations montrent que le PSO+TS obtient de meilleurs résultats que le PSO+SA.

Le *Multi-swarm Particle Swarm Optimization* (MPSO) (Liu et al., 2009) est un algorithme co-évolutionnaire coopératif pour résoudre le FJSP avec les objectifs de minimisation du *makespan*, de la charge de travail sur les machines et sur la machine critique. Deux populations de particules évoluent comme dans un PSO classique, la première contient les informations sur l'ordonnancement et la seconde sur l'affectation des opérations aux machines. Les individus sont évalués en fonction de la qualité de leur collaboration avec ceux de l'autre population. Les résultats montrent que le MPSO est plus performant qu'un PSO classique ou qu'un GA.

L'ABC est une adaptation du PSO pour l'optimisation combinatoire et il existe plusieurs adaptations qui résolvent le FJSP multiobjectif. Par exemple (Zhou et al., 2011), cet MOABC pour le FJSP prend en compte les objectifs de minimisation du *makespan*, de la somme de la charge de travail sur les machines et de la charge sur la machine la plus utilisée. Lors de la phase *Employed Bee*, des opérateurs de croisement sont appliqués sur les parties ordonnancement (MPOX) et affectation (deux points et uniforme) des sources de nourriture. Un opérateur de mutation est appliqué sur la partie affectation par le changement aléatoire d'un gène. Dans la phase *Onlooker Bee*, pour chaque solution le chemin critique est identifié et le déplacement des opérations le composant sert de voisinage pour une recherche locale appliquée par la suite. La phase *Scoot Bee*, des solutions sont générées aléatoirement pour améliorer et diversifier la population. Cet MOABC surpasse les performances du AL-CGA et du PSO+TS.

Le *Enhanced Pareto based Artificial Bee Colony* (EPABC) (Wang et al., 2012a) est un autre MOABC utilisé pour le FJSP avec des objectifs identiques. Par contre, les procédures utilisées dans les phases sont différentes. Dans la phase *Employed Bee*, le voisinage des sources de nourriture sont explorées aléatoirement pour tenter de les améliorer. La phase *Onlooker Bee* est composée de trois étapes. La première est la sélection par tournoi binaire de solutions sur lesquelles est appliquée la même procédure que dans la phase *Scoot Bee*. L'étape suivante utilise les croisements MPOX et ASX pour créer de nouvelles solutions. Et dans la dernière, une recherche locale est appliquée sur le chemin critique de chaque source de nourriture. À la fin de la phase, les meilleures solutions sont retenues pour constituer la nouvelle population. Dans la phase *Scout Bee*, des solutions non-dominées sont sélectionnées aléatoirement et améliorées par une recherche locale sur le chemin critique pour remplacer les plus mauvaises solutions de la population.

Le *Pareto-based Discrete Artificial Bee Colony* (PDABC) (Li et al., 2011a) garde la structure algorithmique d'un MOABC. Les objectifs sont identiques à ceux du EPABC, mais les procédures utilisées dans les phases sont différentes. Dans la phase *Employed Bee*, une recherche locale est faite sur toutes les sources de nourriture pour tenter de les améliorer. Un croisement à un point est ensuite effectué pour obtenir de nouvelles les solutions, les meilleures solutions sont gardées dans la population. Dans la phase *Onlooker Bee* des sources de nourriture sont sélectionnées par tournoi binaire et une recherche locale leur est appliquées. Dans la phase *Scoot Bee* pour assurer une diversification de bonne qualité, des solutions non-dominées sont sélectionnées et modifiées aléatoirement pour ensuite remplacer des sources de nourriture dans la population courante. À la fin de chaque phase l'archive contenant les sources de nourriture non-dominées est mise à jour.

Le PDABC inspire d'autre MOABC pour résoudre le MOFJSP, tel que le *Hybrid Artificial Bee Colony* (Li et al., 2011c). La principale différence vient de la suppression du croisement des solutions dans la phase *Employed Bee*.

Les métheuristiques avec un seul individu proposées pour le FJSP multiobjectif sont peu nombreuses, mais quelques exemples existent. Par exemple, la recherche locale est utilisée pour résoudre le FJSP multiobjectif (Xing et al., 2009a). Le fonctionnement de cet algorithme est assez simple, un ensemble de solutions réalisables est créé à chaque itération. La meilleure solution est sélectionnée. Une LS est effectuée sur cette solution. À chaque itération le voisinage de la solution est explorée, évaluant tous les mouvements vers une solution réalisable. Le meilleur mouvement est gardé pour la prochaine itération. La LS est répétée jusqu'à atteindre le critère d'arrêt.

Fonctionnant sur le même principe, le *Hybrid Tabu Search Algorithm* (HTSA) (Li et al., 2010) est un MOTS conçu pour résoudre le FJSP avec les objectifs de minimisation du *makespan*, de la somme de la charge de travail sur les machines et de la charge sur la machine la plus utilisée. Le HTSA est constitué d'une phase "initialisation" et d'une phase "évolution". Lors de la phase initialisation, une population est créée et chaque solution est évaluée. La meilleure solution est choisie pour la phase d'évolution. Lors de la phase d'évolution, une recherche locale est faite dans la partie affectation des opérations aux machines et ensuite dans l'ordonnancement des opérations.

Quelques travaux existent sur la production des lots juste à temps pour le FJSP. Un *Multiagent Scheduling Method* résout le FJSP avec l'objectif de diminuer la somme des *Tardiness* et des *Earliness* (Wu and Weng, 2005). Cet algorithme est composé de deux types d'agents. Le *Job Agent* (JA) qui choisit les machines pour réaliser les opérations des lots. Le *Machine Agent* (MA) qui ordonnance les opérations sur les machines. Les JA et MA collaborent pour construire une solution. Cette méthode fournit une bonne solution en un temps raisonnable pour une instance de 10 machines avec 2000 lots à ordonnancer.

Avec un bon système de voisinage, un TS est efficace pour résoudre le FJSP avec l'objectif de produire les lots juste à temps (Imanipour and Zegordi, 2006). Une procédure de reconstruction est aussi proposée. Les simulations montrent que le TS donne de bonnes solutions en un temps

raisonnables.

Le FJSP a beaucoup d'application dans l'industrie. Le *Distributed-Intelligence Approaches* (Weng and Fujimura, 2010) est une heuristique qui permet de modifier dynamiquement un ordonnancement pour produire les lots à temps tout en minimisant les stocks. Les expérimentations dans deux environnements industriels donnent des résultats prometteurs.

Le *Two PHeromones Colony Optimization* (2PH-ACO) résout le FJSP en minimisant la somme des *Tardiness* et des *Earliness* (Huang et al., 2013). Le principe est de réduire les coûts environnementaux en réduisant le stockage tout en réalisant les lots pour leur dates d'échéances. Lors des simulations le 2PH-ACO surclasse un ACO classique.

## 3.3 Adaptation de la MCE à l'optimisation multiobjectif

Cette section présente l'adaptation de la MCE aux problèmes multiobjectifs. Le FJSP est gardé comme problème de référence et un objectif est ajouté pour le rendre multiobjectif. Plusieurs méthodes, pour résoudre le FJSP multiobjectif, sont présentés pour être comparées entre elles sans et avec l'hybridation par la MCE. L'hybridation par la MCE garde le même principe que pour l'optimisation mono-objectif, mais la meilleure solution est remplacée par l'approximation du front de Pareto optimal.

### 3.3.1 *Multi-Objective Flexible Job Shop Problem* (MOFJSP)

Le FJSP est gardé comme problème de référence en ajoutant un deuxième objectif la production juste à temps des lots. Pour rappel, résoudre un FJSP consiste à organiser l'exécution de $n$ lots, noté $J = \{J_1, ..., J_n\}$ où chaque lot $J_i$ est composé de suite ordonnée de $h_i$ opérations sur un ensemble de $m$ machines, noté $M = \{M_1, M_2, ..., M_m\}$. La gamme opératoire d'un lot $J_i$ est constituée des opérations $O_{i,1}, O_{i,2}, ..., O_{i,h_i}$ à réaliser dans l'ordre. Pour chaque opération $O_{i,j}$, un ensemble de machines $M_{i,j} \subset M$ est qualifié pour l'exécuter.

Les hypothèses suivantes sont considérées : (1) les lots sont indépendants les uns des autres, (2) les machines sont indépendantes les unes des autres, (3) le temps de préparation des machines est considéré comme négligeable, (4) le temps de déplacement des lots entre les machines est négligé, (5) une machine n'exécute qu'une opération à la fois, (6) une et seulement une opération sur un lot peut-être en cours de réalisation et (7) les opérations de lots différents n'ont pas de contrainte de précédence entre elles.

Les deux objectifs considérés sont la minimisation du maximum des dates de fin de réalisation des lots, le *makespan* ($C_{max}$) (Équation 3.1) et la production des lots juste à temps pour une date d'échéance donnée (Équation 3.2). L'objectif de production juste à temps consiste à additionner la somme des retards (*Tardiness* $T_i$) et des avances (*Earliness* $E_i$) de leur date de fin de réalisation par

rapport à leur date d'échéance (en anglais *due date*) $d_i$. Les objectifs sont définis par les équations suivantes :

$$C_{max} = max\{c_i | i = 1, ..., n\} \tag{3.1}$$

$$\min \sum_{i=1}^{n} (E_i + T_i) \tag{3.2}$$

avec $c_i$ la date de fin de réalisation du lot $i$. Les variables $E_i$ et $T_i$ sont définies par $E_i = max\{0, d_i - c_i\}$ et $T_i = max\{0, c_i - d_i\}$.

La date d'échéance $d_i$ d'un lot $i$ est calculée par une formule (voir équation 3.3). Cette formule permet à l'utilisateur de choisir une date d'échéance cohérente pour tous les lots.

$$d_i = \sum_{j=1}^{h_i} \frac{\sum\limits_{k \in M_{ij}} p_{i,j}^k}{|M_{ij}|} \times cd \tag{3.3}$$

avec $p_{i,j}^k$ le temps de réalisation de l'opération $O_{i,j}$ sur la machine $k$ et $cd$ le coefficient pour la date d'échéance.

En reprenant l'exemple de la section 2.8.1, le tableau 3.1 présente l'instance modifiée avec les dates d'échéance des lots. La dernière ligne du tableau présente les dates d'échéance calculées par l'équation 3.3. L'instance présentée contient 3 lots chacun avec une suite de 3 opérations, soit un total de 9 opérations à ordonner sur 3 machines. Le coefficient de la date d'échéance est $cd = 1.0$.

TABLEAU 3.1 – Exemple d'instance (3/3/3) avec des dates d'échéance

| | Lot 1 | | | Lot 2 | | | Lot 3 | | |
|---|---|---|---|---|---|---|---|---|---|
| | $O_{11}$ | $O_{12}$ | $O_{13}$ | $O_{21}$ | $O_{22}$ | $O_{23}$ | $O_{31}$ | $O_{32}$ | $O_{33}$ |
| Temps $M_1$ | 1 | - | - | - | 3 | 3 | 4 | - | - |
| Temps $M_2$ | - | 4 | - | 1 | - | - | - | 1 | 5 |
| Temps $M_3$ | - | 4 | 2 | - | 3 | 3 | 4 | - | - |
| *Due date* | | 7 | | | 7 | | | 10 | |

### 3.3.2 Algorithmes évolutionnaires et apparentés

Cette section présente les adaptations implémentées du NSGAII, du SPEA2 et du MOAIS pour résoudre le FJSP multiobjectif. À la fin, le fonctionnement de ces opérateurs sont décrits par la suite.

#### 3.3.2.1 *Non-dominated Sorting Algorithm II* (NSGAII)

Le NSGA-II (Deb et al., 2002) est l'un des plus célèbres algorithmes évolutionnaires pour l'optimisation multiobjectif. Il est simple d'utilisation et obtient de bons résultats. La section 1.4.1 en décrit le fonctionnement. Le chromosome utilisé est l'un des plus efficace de la littérature (Tay and Wibowo, 2004) pour les représentations de solutions du FJSP. Les solutions sont évaluées et ensuite classées par rang de Pareto. Pour assurer la diversité, les solutions d'un front de Pareto sont évaluées en fonction du *crowding distance*. Le *crowding distance* privilégiant la sélection des notes des extrêmes, le NSGAII étire ses fronts de Pareto assurant ainsi une bonne diversité des solutions non-dominées. Les opérateurs utilisés lors de la reproduction sont la roulette *wheel* pour la sélection des parents (voir figure 3.1), le croisement à un point (voir figure 3.2) et pour la mutation l'échange de positions de deux gènes et la modification de la valeur d'un gène (voir figure 3.3).

#### 3.3.2.2 *Strength Pareto Evolutinary Algorithm 2* (SPEA2)

Le SPEA2 (Zitzler et al., 2001) est un algorithme évolutionnaire très commun, souvent adapté à de nombreux problèmes d'optimisation multiobjectif. Son fonctionnement est décrit dans la section 1.4.1. À chaque itération, il suit la trame classique d'un EA : sélection des parents, reproduction (croisement et mutation) et survie des meilleurs individus. Le SPEA2 se caractérise par l'utilisation d'une archive pour stocker les solutions non-dominées et de la technique d'évaluation du *fine-grained*. Pour l'adapter au FJSP, le génotype ou chromosome (Tay and Wibowo, 2004) est composé de deux parties, l'une désignant une machine pour chaque opération et l'autre l'ordre de passage des opérations sur les machines. Les parents sont sélectionnés par un système de roulette *wheel* (voir figure 3.1). Le croisement utilisé est un croisement à un point (voir figure 3.2). La mutation est un échange (ou en anglais *swap*) où deux gènes sont intervertis dans la partie ordonnancement. Dans la partie affectation, la mutation change la valeur d'un gène (voir figure 3.3).

#### 3.3.2.3 *Multi-Objective Artificial Immune System* (MOAIS)

L'*Artificial Immune System* AIS est inspiré du système immunitaire qui défend un organisme contre l'intrusion de corps étrangers. Des anticorps sont produits par l'organisme pour lutter contre les pathogènes. Le système immunitaire identifie une intrusion par des virus, bactéries, etc ... Des anticorps sont produits pour lutter contre cette intrusion. Ces anticorps cherchent les pathogènes dans l'organisme. Quand un pathogène est trouvé, il est immédiatement attaqué pour être détruit. L'efficacité d'un anticorps dans la recherche et la lutte contre un pathogène dépend de leurs caractéristiques. En fonction des résultats obtenus, l'organisme modifie les caractéristiques et la production des anticorps. Ainsi la lutte contre les pathogènes gagne en efficacité.

Le MOAIS implémenté est inspiré de l'AIA (Bagheri et al., 2010) et du STCSA (Lu, 2009) qui sont deux algorithmes conçus pour résoudre le FJSP. Dans l'AIS, les individus de la population sont

considérés comme les anticorps ou antibiotiques et la fonction objectif comme les pathogènes. Le FJSP est divisible en deux sous problèmes : le premier consiste à affecter les opérations aux machines et le second à les ordonnées sur les machines. La représentation d'un antibiotique est donc composée de deux parties l'une pour l'affectation des opérations aux machines et la seconde pour l'ordonnancement des opérations sur les machines (Tay and Wibowo, 2004). La trame d'un MOAIS est très simple. Dans l'initialisation, une population d'individus est créée par des heuristiques pour garantir la bonne qualité et la diversité des solutions. À chaque itération, des individus sont sélectionnés au hasard pour être clonés, les meilleurs individus ont plus de probabilité d'être choisis. Après le clonage, les individus se modifient par hypermutation (voir Figure 3.3), ces individus modifiés sont ensuite évalués avec la fonction objectif. Les meilleurs individus parmi la population courante et les nouveaux sont sélectionnés pour constituer la population de l'itération suivante.

**Étape 1 :** Initialisation de la population et évaluation des individus.

**Étape 2 :** Sélection par roulette *wheel* et clonage d'individus dans la population.

**Étape 3 :** Hypermutation des individus clonés.

**Étape 4 :** Évaluation des individus modifiés.

**Étape 5 :** Classement par rang de Pareto des nouvelles solutions et celles dans la population.

**Étape 6 :** Les individus les mieux classés sont retenus pour constituer la prochaine population.

**Étape 7 :** Si le critère d'arrêt est atteint, retourner les solutions non-dominées de la population, sinon retour à l'étape 2.

#### 3.3.2.4 Opérateurs évolutionnaires

Cette section décrit les opérateurs utilisés par les algorithmes évolutionnaires et apparentés. Certains algorithmes décrits par la suite utilisent certains de ces opérateurs. Les opérateurs présentés sont : la sélection par roulette *Wheel*, le croisement à un point et la mutation pour le FJSP.

**Opérateur de sélection**

La sélection par roulette *Wheel* est utilisé pour sélectionner les parents. Conceptuellement, à chaque individu de la population est alloué un intervalle de la roulette *Wheel*. Comme dans une roulette *Wheel* réelle, les intervalles ont différentes tailles, proportionnels à l'évaluation de l'individu. Le meilleur individu possède la plus grande part et le plus mauvais la plus petite.

Le tableau 3.2 et la Figure 3.1 contiennent un exemple de population et la roulette *Wheel* correspondante utilisée pour la sélection des parents. Dans le cas d'une optimisation multiobjectif, l'évaluation correspond au rang de Pareto. Un nombre aléatoire est généré, l'individu correspondant à l'intervalle auquel appartient ce nombre est sélectionné. L'opération est refaite jusqu'à obtenir le nombre de parents voulu pour créer la nouvelle génération. Remarque, un individu peut être sélectionné plusieurs fois comme parent.

TABLEAU 3.2 – Exemple de population avec son évaluation

| Individu | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Rang de Pareto | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 4 |
| Probabilité | 0,133 | 0,133 | 0,133 | 0,133 | 0,100 | 0,100 | 0,100 | 0,066 | 0,066 | 0,033 |

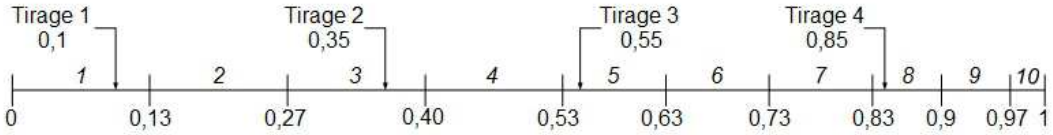


FIGURE 3.1 – Roulette Wheel correspondant à la population du tableau 3.2 avec quatre tirages

**Opérateur de croisement**

L'opérateur de croisement à un point est l'un des plus utilisés par les algorithmes génétiques. Le principe consiste à utiliser le génotype de deux parents en choisissant deux individus dans la population courante. Le génotype de chaque parent est découpé en deux morceaux exactement au même point de coupure. Pour obtenir deux nouveaux individus, la première partie d'un génotype est réassemblée avec la deuxième partie de l'autre génotype.

Le chromosome du FJSP est composé de deux parties (Tay and Wibowo, 2004), l'une pour ordonnancement et l'autre pour l'affectation des opérations sur les machines. Chaque partie a le même nombre de gènes qui est égal au nombre d'opérations. Le croisement à un point est appliqué à chaque partie du chromosome (voir Figure 3.2). Une procédure différente de croisement est effectuée pour chaque partie du génotype. Les deux procédures sont différentes et adaptées à la nature de chacune des deux parties. Elles ont comme caractéristique commune d'obtenir des enfants réalisables sans passer par une procédure de réparation.


FIGURE 3.2 – Croisement à un point pour le FJSP

Pour la partie "séquence des opérations" sur les machines, les étapes de l'opérateur de croisement sont les suivantes :

**Étape 1 :** Générer un nombre aléatoire $p$, tel que $p \in [2, n-1]$, avec $n$ le nombre d'opérations dans l'instance.

**Étape 2 :** Sélectionner la sous-partie avant $p$ dans les parents 1 et 2.

**Étape 3 :** Copier respectivement ces deux sous-parties dans les enfants 1 et 2 en respectant les positions.

**Étape 4 :** Deux vecteurs intermédiaires 1 et 2 sont créés, en enlevant respectivement dans les parents 1 et 2 les gènes déjà présents dans les enfants 2 et 1.

**Étape 5 :** Compléter respectivement les enfants 1 et 2 en ajoutant dans l'ordre les gènes des vecteurs 2 et 1 aux places libres.

Pour la partie "affectation des opérations" aux machines, la procédure de croisement s'effectue comme suit :

**Étape 1 :** Générer un nombre aléatoire $p$, tel que $p \in [2, n-1]$, avec $n$ le nombre d'opérations dans l'instance.

**Étape 2 :** Sélectionner la sous-partie avant $p$ dans les parents 1 et 2.

**Étape 3 :** Copier respectivement ces deux sous-parties dans les enfants 1 et 2 en respectant les positions.

**Étape 4 :** Sélectionner la sous-partie après $p$ dans les parents 1 et 2.

**Étape 5 :** Copier respectivement les sous-parties des parents 1 et 2 dans les enfants 2 et 1 en suivant les positions des gènes.

**Exemple 2** (Opérateur de croisement à un point)**.** *Cet exemple décrit la procédure de la Figure 3.2. Le chromosome du premier parent est* $\{1,2,1,2,0,1,0,2\}$ *et* $\{0,1,0,0,2,1,2,0\}$*, celui du deuxième est* $\{2,0,1,0,2,1,1,2\}$ *et* $\{1,1,2,1,0,2,0,1\}$*. Le point de croisement p est* 4*.*

*Pour la partie "Ordre des opérations", le vecteur venant du parent 1* $\{1,2,1\}$ *est copié dans l'enfant 1. Le vecteur du parent 2 est* $\{2,0,1,0,2,1,1,2\}$*, après avoir enlevé les opérations déjà présentes dans l'enfant 1, le vecteur intermédiaire obtenu est* $\{0,0,2,1,2\}$*.*

*Pour constituer l'enfant 2, la partie du parent 2 est* $\{2,0,1\}$*, le vecteur venant du parent 1 est* $\{1,2,1,0,2\}$*. La partie "Ordre des opérations" de l'enfant 2 est* $\{2,0,1,1,2,1,0,2\}$*.*

*Pour la partie "affectation des opérations", l'enfant 1 est constitué des vecteurs* $\{0,1,0\}$ *et* $\{0,2,1,2,0\}$ *venant respectivement des parents 1 et 2. Le résultat est* $\{0,1,0,0,2,1,2,0\}$ *pour l'enfant 1. Les parents 1 et 2 donnent les vecteurs* $\{1,1,2\}$ *et* $\{0,2,1,2,0\}$ *pour construire la partie "Affectation des opérations" de l'enfant 2* $\{1,1,2,0,2,1,2,0\}$*.*

**Opérateur de mutation**

La mutation concerne un seul individu, et même un seul gène du génotype qui est modifié. L'individu mute et la solution qu'il représente est modifiée obtenant de nouveaux résultats de la fonction objectif.

Pour le FJSP, la mutation est effectuée sur les deux parties du chromosome (voir Figure 3.3). Généralement pour chaque gène du chromosome, un test aléatoire détermine si le gène doit muter. La probabilité de la mutation d'un gène est de $1/n$, avec $n$ le nombre de gènes dans le chromosome.

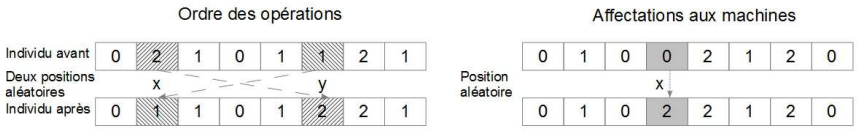


FIGURE 3.3 – Opérateur de mutation pour le FJSP

Pour la partie "ordonnancement des opérations" sur les machines, la procédure de mutation intervertit la place de deux gènes. La mutation pour chaque gène suit les étapes suivantes :

**Étape 1 :** Pour le gène à la position $x$, générer un nombre aléatoire $p \in [0, 1]$.

**Étape 2 :** Si $p > \frac{1}{n}$, retour à l'étape 1 et passer au gène suivant ($x = x + 1$), sinon aller à l'étape suivante.

**Étape 3 :** Choisir une position aléatoire $y$, tel que $y \in [1, n]/\{x\}$.

**Étape 4 :** Échanger la place des deux gènes, le gène à la position $x$ prend la place $y$ et inversement.

Pour la partie "affectation des opérations" aux machines, la valeur d'un gène est modifiée. Les étapes de cette mutation sont décrites ci-dessous :

**Étape 1 :** Pour l'opération $O_{ij}$ à la position $x$, générer un réel aléatoire $p \in [0, 1]$.

**Étape 2 :** Si $p > 1/n$ $x = x + 1$ et retour à l'étape 1, sinon continuer.

**Étape 3 :** Choisir une valeur $m_2 \in \{0, |M_{ij}| - 1\}/\{m_1\}$, avec $m_1$ la valeur courante à la position $x$ et $|M_{ij}|$ est le nombre de machines dans l'ensemble $M_{ij}$ et remplacer $m_1$ par $m_2$.

**Exemple 3** (Opérateur de mutation)**.** *Cet exemple décrit la procédure de la Figure 3.3. Le génotype de l'individu est* $\{0, 2, 1, 0, 1, 1, 2, 1\}$ *et* $\{0, 1, 0, 0, 2, 1, 2, 0\}$.

*Pour la partie "Ordre des opérations", les positions* $x$ *et* $y$ *sont* 2 *et* 6. *Les gènes avec les valeurs* 2 *et* 1 *échangent leur position. Le nouveau génotype est* $\{0, 1, 1, 0, 1, 2, 2, 1\}$.

*Pour la partie "Affectation des opérations", le gène avec la valeur* 0 *à la position* $x = 4$. *La nouvelle valeur du gène est* 2, *donnant le nouveau chromosome* $\{0, 1, 0, 2, 2, 1, 2, 0\}$.

### 3.3.3 *Multi-Objective Ant Colony Optimization* (MOACO)

Le MOACO est une adaptation de deux algorithmes, le *Knowledge-Based Ant Colony Optimization* (KBACO) (Xing et al., 2010) et le *Double Layer Ant Colony Optimization* (DLACO)(Xing et al., 2008). Le premier est conçu pour résoudre le FJSP mono-objectif et le second pour le FJSP multi-objectif. L'ACO imite le déplacement des fourmis pour ramener de la nourriture, les fourmis identifient le chemin le plus court grâce au dépôt de phéromones de leurs congénères ayant déjà fait le chemin. Les phéromones s'évaporent régulièrement donc les chemins les plus courts ont plus de chance d'être choisis par les fourmis pour rejoindre leur destination.

Le MOACO possède donc trois ensembles, les fourmis qui construisent des solutions à chaque itération, les mémoires ou dépôt de phéromones pour retenir la place des meilleures positions et une archive pour stocker les solutions non-dominées. Cet MOACO a une seule colonie avec une seule espèce de fourmis. Les mémoires sont mises à jour à chaque itération ; sur chaque position se trouve des phéromones déposées lors des itérations précédentes. La mise à jour se fait en deux étapes. La première, l'évaporation, concernant toutes les positions, chacune perd une petite quantité de phéromones. La seconde, le dépôt, où les meilleures solutions sont identifiées et des phéromones sont déposées sur chacune de leurs positions. À l'itération suivante, les fourmis construisent de nouvelles solutions en choisissant les positions au hasard, mais celles qui ont beaucoup de phéromones ont plus de chance d'être choisies. À chaque itération, l'archive est mise à jour et l'algorithme sauvegarde les solutions non-dominées explorées.

Pour le FJSP, la représentation d'une solution est composée de deux parties, l'une pour l'ordonnancement, l'autre pour l'affectation (Tay and Wibowo, 2004). La mémoire est constituée de deux parties, l'une pour retenir l'ordonnancement des opérations et la seconde pour l'affectation des opérations aux machines.

La mémoire pour l'ordonnancement s'appelle *Operation Assignment Priority Knowledge* (OAPK). OAPK accumule la connaissance pour connaître la meilleure priorité (ou place sur les machines) pour une opération. Ses connaissances sont issues des solutions non-dominées. La mémoire OAPK est une matrice de dimension $n \times h \times m$, où $n$ est le nombre de lots dans l'instance, $h$ le nombre maximum d'opérations par lot et $m$ le nombre de machine. Les priorités sont initialisées avec la même valeur.

La mémoire pour l'affectation se nomme *Operation Assignment Machine Knowledge* (OAMK). OAMK contient les informations sur la meilleure machine pour réaliser une opération donnée. Les meilleures solutions sont utilisées pour collecter les informations pour OAMK. La mémoire OAMK est une matrice de dimension $n \times h \times m$, où $n$ est le nombre de lots dans l'instance, $h$ le nombre maximum d'opérations par lot et $m$ le nombre de machines. La matrice OAMK en fonction de la qualification des machines et du temps de réalisation des opérations (voir équation 3.4). Si une machine n'est pas qualifiée pour une opération, alors la position correspondante prend la valeur 0. Sinon la position prend une valeur inversement proportionnelle au temps de réalisation de l'opération

sur la machine.

$$OAMK(i,j,k) = \begin{cases} 0 & Si\ M_k \notin M_{i,j} \\ 1/p_{i,j}^k & Si\ M_k \in M_{i,j} \end{cases} \tag{3.4}$$

avec $p_{i,j}^k$ le temps de réalisation de l'opération $O_{i,j}$ sur la machine $M_k$.

L'algorithme du MOACO suit la trame générale suivante :

**Étape 1 :** Initialisation des matrices mémoires OAPK et OAMK.

**Étape 2 :** Affectation des opérations aux machines pour chaque fourmi en utilisant la mémoire OAMK.

**Étape 3 :** Ordonnancement des opérations sur les machines avec l'aide de la mémoire OAPK.

**Étape 4 :** Amélioration en créant de nouvelles solutions par croisement.

**Étape 5 :** Évaluation des solutions et mise à jour des solutions dans l'archive.

**Étape 6 :** Mise à jour des mémoires OAPK et OAMK avec les solutions dans l'archive.

**Étape 7 :** Si le critère d'arrêt n'est pas atteint, retour à l'étape 2, sinon arrêt de l'algorithme.

La procédure d'affectation détermine les machines qui exécuteront les opérations. Dans cette procédure, les fourmis affectent, les opérations à des machines qualifiées pour les réaliser. Pour chaque opération, les machines sont choisies au hasard, mais la mémoire OAMK est utilisée pour déterminer la probabilité qu'une machine soit sélectionnée. L'affectation des opérations se fait comme suit :

**Étape 1 :** Sélectionner une opération $O_{i,j}$ qui n'est pas affectée à une machine.

**Étape 2 :** Pour chaque machine qualifiée, la probabilité $Pr(i,j,k)$ d'être choisie est calculée d'après l'équation 3.5.

$$Pr(i,j,k) = \frac{OAMK(i,j,k)}{\sum_{t=1}^{m} OAMK(i,j,t)} \tag{3.5}$$

avec $Pr(i,j,k)$ la probabilité d'affectation de l'opération $O_{i,j}$ à la machine $k$, OAMK(i,j,k) la quantité de phéromones déposées pour affecter l'opération $O_{i,j}$ à la machine $k$.

**Étape 3 :** En respectant les probabilités calculées à l'étape précédente, affecter aléatoirement à une machine.

**Étape 4 :** Si toutes les opérations sont affectées, arrêt de la procédure, sinon retour à la étape 1.

Dans l'étape suivante, les opérations sont ordonnées sur les machines. Après avoir affecté les opérations aux machines, les fourmis ordonnent les opérations sur les machines avec l'aide de la mémoire OPAK. Les opérations sont ordonnancées les unes après les autres sur les machines. Les étapes suivies par une fourmi pour établir cet ordonnancement sont décrites ci-dessous :

**Étape 1 :** Sélectionner la machine $m$ dont les opérations déjà ordonnancées terminent le plus tôt l'instant $t$, si plusieurs candidates existent choisir au hasard.

**Étape 2 :** Constituer l'ensemble des opérations pouvant être ordonnées $disponible(m,t)$. Les opérations candidates sont affectées à la machine $m$ et ne sont pas encore ordonnancées. Les opérations sélectionnées sont celles dont toutes les précédentes sur leur lot sont déjà ordonnancées, ainsi que les opérations sans contrainte de précédence sur le lot.

**Étape 3 :** Parmi cet ensemble, choisir aléatoirement l'opération en fonction de l'équation 3.6.

$$\forall O_{i,j} \in disponible(m,t), \quad Pr(i,j,m,t) = \frac{[OAPK(i,j,m)]^{\alpha} \times \left[1/p_{i,j}^{m}\right]^{\beta}}{\sum_{O_{i,j} \in disponible(m,t)} \left\{[OAPK(i,j,m)]^{\alpha} \times \left[1/p_{i,j}^{m}\right]^{\beta}\right\}} \tag{3.6}$$

Où $Pr(i,j,m,t)$ est la probabilité de sélection de l'opération $O_{i,j}$ sur la machine $m$ à l'instant $t$, $OAPK(i,j,m)$ est la matrice mémoire pour l'ordonnancement, $\alpha$ et $\beta$ sont des coefficients pour régler l'influence des positions dans OAPK et du temps de réalisation $p_{i,j,m}$ de l'opération $O_{i,j}$ sur la mcine $m$.

**Étape 4 :** Tant que toutes les opérations ne sont pas ordonnancées, la fourmi revient à l'étape 1.

La phase amélioration consiste à utiliser un croisement en un point (voir section 3.3.2.4). Après évaluation des fourmis, des solutions sont sélectionnées aléatoirement parmi les non-dominées et ensuite celles de la population courante. Les solutions non-dominées sont croisées avec celles de la population courante. Les nouvelles solutions sont ensuite évaluées et ajoutées à la population courante. Les solutions de la population courante sont ensuite classées par rang de Pareto pour identifier les solutions non-dominées. L'archive est ensuite mise à jour en ne retenant que les solutions non-dominées dans la population courante et l'archive.

Lors de l'étape d'apprentissage, les matrices mémoires OAPK, pour l'ordonnancement, et OAMK, pour l'affectation, sont mises à jour en utilisant les solutions non-dominées. La mise à jour des mémoires utilisée est classique dans les ACOs (Stutzle and Hoos, 2000), des phéromones sont déposées sur les positions des solutions non-dominées et en deuxième temps une petite quantité de phéromones s'évaporent de toutes les positions. Pour maintenir une bonne convergence et diversification de l'exploration de l'espace de solutions, le concept du *Max-Min Ant System* (MMAS). Chaque position des mémoires à une valeur comprise dans l'intervalle $[\tau_{min}, \tau_{max}]$. Le $\tau_{min}$ évite qu'une position ne soit plus utilisée par les fourmis, bloquant l'accès d'une partie de l'espace de recherche à l'algorithme lors de son exploration. Le $\tau_{max}$ empêche qu'une position ne deviennent hégémonique, ce qui peut bloquer l'algorithme dans un minimum local. Les matrices OAMK et OAPK sont mises à jour en deux étapes : le dépôt et l'évaporation des phéromones.

Les règles du dépôt de phéromones sur chaque position suivent les équations 3.7 et 3.8 :

$$OAMK(i,j,k) = OAMK(i,j,k) \times Q_1 \tag{3.7}$$

$$OAPK(i,j,k) = OAPK(i,j,k) + Q_2 \tag{3.8}$$

Où $Q_1$ est le coefficient de dépôt de phéromones pour la matrice OAMK et $Q_2$ celui de la matrice OAPK.

L'évaporation respecte les règles définies par les équations 3.9 et 3.10 :

$$OAMK(i,j,k) = min\{\tau_{max}, max\{\tau_{min}, (1-\rho) \times OAMK(i,j,k)\}\} \tag{3.9}$$

$$OAPK(i,j,k) = min\{\tau_{max}, max\{\tau_{min}, (1-\rho) \times OAPK(i,j,k)\}\} \tag{3.10}$$

Où $\rho$ est le coefficient d'évaporation des phéromones pour l'algorithme. En plus, $OAMK(i,j,k)$ représente la position de l'opération $O_{i,j}$ affectée à la machine $k$ et $OAPK(i,j,k)$ désigne la priorité $k$ de l'opération $O_{i,j}$.

### 3.3.4 *Multi-Objective Particle Swarm Optimization* (MOPSO)

Le *Particle Swarm Optimization* (PSO) est un algorithme à population (Kennedy and Eberhart, 1995) pour l'optimisation continue. Le *Multi-Objective Particle Swarm Optimization* (MOPSO) (Coello Coello and Lechuga, 2002) est l'adaptation du PSO pour les problèmes multiobjectifs. Le MOPSO s'adapte au problème combinatoire du FJSP (Moslehi and Mahnam, 2011).

Le PSO à deux caractéristiques, son optimisation par une évolution sociale et sa simplicité d'utilisation. Le PSO requiert des opérateurs mathématiques simples qui consomment peu de mémoires et de temps de calcul. Les individus d'un PSO peuvent être considérés comme des agents cherchant à travers l'espace de solutions. Chaque individu (particule) est, comme un oiseau, dans un espace de recherche multidimensionnel. Toutes les particules sont évaluées par une fonction objectif. La vélocité dirige le vol de la particule. Initialement, le PSO produit une population avec des heuristiques pour bien répartir la population dans l'espace de recherche et attribue des valeurs aléatoires à la vélocité. La vélocité est ajustée dynamiquement à chaque itération selon l'expérience de l'individu et celle du reste de la population (Équation 3.11). Le premier terme de l'équation 3.11 définit le poids de l'inertie de la particule dans la vélocité. Le second terme correspond à l'expérience individuelle de la particule. Le troisième terme est la somme des connaissances de la population. La nouvelle position de la particule est déterminée par l'équation 3.12 à partir de la position courante :

$$V_{i,t+1} = w.V_{i,t} + Rand.C_1.(P_i - X_{i,t}) + rand.C_2.(P_g - X_{i,t}) \tag{3.11}$$

$$X_{i,t+1} = X_{i,t} + V_{i,t+1} \tag{3.12}$$

avec $i$ l'indice de la particule, $X_{i,t}$ est la position de la particule $i$ à l'itération $t$, $V_{i,t}$ est la vélocité de la particule $i$ à l'instant $t$, $P_i$ est la meilleure solution explorée par la particule $i$ (*pbest*) et $P_g$

est la meilleure solution explorée par la population (*gbest*). $w$ est le poids de l'inertie qui détermine l'influence de la trajectoire précédente dans le déplacement. L'inertie décroit linéairement durant l'exécution de l'algorithme. Le poids de l'inertie est recalculé à chaque itération (voir équation 3.13).

$$w = w_{max} - ((w_{max} - w_{min})/t_{max}) \times t \tag{3.13}$$

$w_{max}$ est le coefficient initial de l'inertie, $w_{min}$ la valeur finale de l'inertie, $t_{max}$ le nombre total d'itérations de l'algorithme et $t$ le numéro de l'itération courante. $C_1$ et $C_2$ sont les deux constantes qui définissent respectivement l'influence du *pbest* et du *gbest* dans le calcul de la vélocité de la particule. *Rand* et *rand* sont deux réels aléatoires tirés dans l'intervalle [0,1] à chaque itération complétant l'influence du *pbest* et du *gbest*.

L'algorithme du PSO suit la trame suivante :

**Étape 1 :** Initialisation de la population des particules et de leur vélocité.

**Étape 2 :** Évaluation des particules, initialisation du *pbest* pour chaque particule et identification du *gbest*.

**Étape 3 :** Mise à jour de l'inertie, des positions et des vélocités en fonction des équations 3.11, 3.12 et 3.13.

**Étape 4 :** Évaluation de chaque particule par la fonction objectif.

**Étape 5 :** Pour chaque particule, comparer la solution courante avec le *pbest*, si elle est meilleure alors elle remplace le *pbest*.

**Étape 6 :** Identifier la meilleure particule de la population courante et la comparer avec le *gbest*. Si cette nouvelle solution est meilleure que le *gbest*, alors elle le remplace.

**Étape 7 :** Si le critère est atteint alors le *gbest* est retourné, sinon retour à l'étape 3.

Dans le cas d'un MOPSO, le *gbest* est remplacé par le front de Pareto optimal.

#### 3.3.4.1 Choix du minimum global (*pbest*) et du minimum local (*pbest*) pour le MOPSO

Dans le PSO le choix du meilleur optimum local *pbest* et meilleur optimum global *gbest* se fait facilement, étant donné que les choix possibles sont généralement réduis à une seule solution. La transformation du PSO en MOPSO nécessite une redéfinition de la sélection de l'optimum global (*gbest*).

Dans le cas du PSO, la meilleure solution est choisie comme *gbest*, mais pour le MOPSO le *gbest* d'une particule provient du front de Pareto optimal. Ce problème est important et difficile car il faut garantir une bonne convergence et diversification de la recherche de l'algorithme. Choisir au hasard le *gbest* parmi le front de Pareto ne garantit pas une bonne convergence du MOPSO. Car, pour

assurer une bonne convergence, la procédure de sélection du *gbest* dans l'archive des solutions non-dominées est primordiale pour obtenir le meilleur guide possible pour une particule. Une stratégie élitiste doit être utilisée par le MOPSO qui évalue les particules de la population et les compare avec celles de l'archive. L'archive est mise à jour en ajoutant les nouvelles solutions non-dominées et enlevant celles qui ne le sont plus, ainsi l'archive contient le front de Pareto optimal de l'algorithme.

La méthode du sigma est utilisée pour trouver le meilleur *gbest* d'une particule (Mostaghim and Teich, 2003). Selon cette méthode une valeur $\sigma_i$ est assignée à chaque solution avec les coordonnées $(f_{1i}, f_{2i})$ et $\sigma$ est définit par $\sigma = (f_1^2 - f_2^2)/(f_1^2 + f_2^2)$. Ce qui implique que toutes les solutions sur la ligne $f_2 = af1$ ont la même valeur pour $\sigma = (1-a)^2/(1+a)^2$.

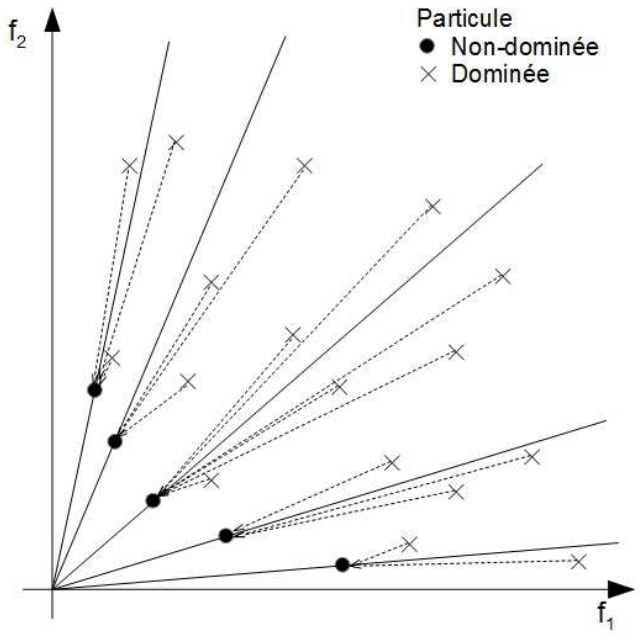


FIGURE 3.4 – Trouver le meilleur *gbest* avec la stratégie élitiste de la méthode du sigma.

Afin de trouver le meilleur *gbest* d'une particule, dans une première étape, la valeur $\sigma_j$ de chaque solution $j$ dans l'archive est déterminée. En deuxième étape, le $\sigma_i$ de chaque particule $i$ de la population est calculé. Alors, la distance entre le $\sigma_i$ et les $\sigma_j$ des solutions dans l'archive est calculée pour chaque particule. Finalement, une particule choisit comme *gbest* la solution dans l'archive dont le $\sigma_j$ à la plus petite distance avec son $\sigma_i$. En d'autres termes, une particule choisit la solution non-dominée la plus proche comme *gbest*.

Le meilleur optimum local (*pbest*) assure la diversification dans le déplacement d'une particule. Dans le PSO, la solution avec la meilleure évaluation parmi celles qui ont été explorées est choisie comme *pbest*. Le *pbest* est second point d'attraction dans le déplacement d'un particule. Cette influence permet à la particule de tomber dans un minimum local. Dans le MOPSO, le *pbest* de

chaque particule est mis à jour à chaque itération. Si la nouvelle solution explorée par la particule domine strictement le *pbest*, elle le remplace. Le *pbest* reste inchangé dans le cas inverse. Cependant, si les deux solutions sont non-dominées entre elles, le choix se fait aléatoirement.

#### 3.3.4.2 Représentation de la particule pour le FJSP

Le plus important pour utiliser un PSO résolvant le FJSP est de développer un mécanisme efficace de fabrication de nouvelles solutions. La particule du PSO est composée de deux parties, l'une encodant l'ordonnancement des opérations et la seconde contenant les affectations des opérations aux machines.

Pour la partie ordonnancement, le codage exploite une clé aléatoire utilisée U(0,1) par (Goncalves et al., 2005) et une stratégie identique à celle proposée par (Bean, 1994). L'avantage est que chaque codage correspond à une solution réalisable. Tout vecteur de clés peut être interprété comme une solution faisable, ce qui implique que tout déplacement particule est possible pour la particule.

La partie ordonnancement est encodée par une suite de clés aléatoires (réels aléatoires compris entre 0 et 1). Elle est composée de $2n$ gènes, où $n$ est le nombre d'opérations.

$Partie\ Ordonnancement\ =\ (x_1, x_2, ..., x_n, x_{n+1}, ..., x_{2n})$

Les $n$ premiers gènes sont les priorités des opérations :

$Priorité_j = x_j$

Les gènes entre $n+1$ et $2n$ sont utilisés pour déterminer les temps de délai utilisés pour calculer l'ordonnancement des opérations. Les temps de délai sont utilisés à chaque itération $g$, $Délai_g$, est calculé par l'expression suivante :

$Délai_g = x_g \times 1,5 \times d_{max}$

Où $d_{max}$ est la durée maximum de toutes les opérations.

Pour la partie affectation, un vecteur de position indiquant l'affectation de chaque opération à une machine qualifiée (Xia and Wu, 2005). Sur le vecteur, les opérations sont rangées en fonction du numéro de leur lot et de leur place dans la gamme opératoire. Premièrement, pour toutes les opérations dans chaque instance, les machines capables de faire une opération sont triées par ordre croissant selon leur temps de réalisation. En fonction de ce classement, une priorité est attribuée à la machine pour l'opération. La machine la mieux classée a la priorité 1, et ainsi de suite pour les suivantes. La position d'une opération correspond à la priorité d'une machine qualifiée. L'opération est affectée à la machine à laquelle correspond cette priorité. Quand la particule se déplace la position ne peut pas excéder le nombre de machines qualifiées, ni descendre en dessous de 1.

Les tableaux 3.3 et 3.4 montrent un exemple d'affectation d'opérations aux machines. Le tableau 3.3 présente l'instance et le classement des machines en fonction de leur temps de réalisation pour chaque opération. L'instance est constituée de deux lots à ordonnancer sur quatre machines. La

partie gauche du tableau contient la description des lots des machines qualifiées et des temps de réalisation. Dans la partie droite, se trouvent les machines classées en fonction de leur priorité. Le tableau 3.4 propose un exemple de vecteur de position d'une particule, représentant l'affectation des opérations aux machines.

TABLEAU 3.3 – Exemple de problème

| | | Machine | | | | Ordre de priorité | | |
|---|---|---|---|---|---|---|---|---|
| | | $M_1$ | $M_2$ | $M_3$ | $M_4$ | 1 | 2 | 3 |
| $J_1$ | $O_{11}$ | 2 | - | 4 | 1 | $M_4$ | $M_1$ | $M_3$ |
| | $O_{12}$ | - | 1 | 8 | 2 | $M_2$ | $M_4$ | $M_3$ |
| $J_2$ | $O_{21}$ | 1 | 4 | - | 2 | $M_1$ | $M_4$ | $M_2$ |
| | $O_{22}$ | 5 | 3 | 2 | - | $M_3$ | $M_2$ | $M_1$ |
| | $O_{23}$ | 3 | - | 1 | 4 | $M_3$ | $M_1$ | $M_4$ |

TABLEAU 3.4 – Exemple d'affectation des opérations aux machines

| Lot | $J_1$ | | $J_2$ | | |
|---|---|---|---|---|---|
| Opération | $O_{11}$ | $O_{12}$ | $O_{21}$ | $O_{22}$ | $O_{23}$ |
| Position de la particule | 2 | 1 | 3 | 2 | 3 |
| | ↓ | ↓ | ↓ | ↓ | ↓ |
| Temps de réalisation | $M_1$ | $M_2$ | $M_2$ | $M_2$ | $M_4$ |

En général, les positions et les vélocités d'une particule sont générées aléatoirement. Lors de l'initialisation pour améliorer la qualité des solutions obtenues, l'espace de recherche peut être réduit pour obtenir des positions initiales de bonne qualité. Par exemple, les machines avec la plus mauvaise priorité peuvent être exclues lors de la construction de solutions initiales.

Chaque position de particule représente la priorité pour chaque opération. Chaque bit de vecteur de position doit être un entier. Mais après les calculs par les équations 3.12 et 3.13, la position des particules est réelle. La priorité de l'opération est égale à la partie entière inférieure de la position. Ainsi, un algorithme conçu pour l'optimisation continue est adapté au problème combinatoire du FJSP.

**Procédure d'ordonnancement des opérations sur les machines**

La particule est composée de deux parties, chacune représentant l'un des sous-problèmes du FJSP, la première, l'ordonnancement et la seconde, l'affectation aux machines. Le décryptage de la particule se fait en deux temps. En premier lieu et présenté ci-dessus, l'affectation des opérations aux machines est retrouvée. Ensuite, l'algorithme 3.1 (Goncalves et al., 2005) est utilisé pour déterminer l'ordonnancement des opérations aux machines. Le principe est de contrôler le temps de délai donné à chaque opération. Par la gestion du maximum du temps de délai autorisé, la taille de l'espace de

recherche est contrôlée. Un temps de délai égal à zéro entraîne une absence de temps d'attente entre les opérations dans l'ordonnancement. L'algorithme 3.1 est basé sur le schéma de l'incrémentation à chaque itération. Chaque partie de l'ordonnancement des opérations étant composée de $2n$ gènes réels dans le domaine $[0, 1]$, où $n$ est le nombre d'opérations dans l'instance. Les $n$ premiers gènes correspondent aux priorités des opérations et les $n$ gènes suivants sont utilisés pour le calcul des temps de délai pour chaque opération. Le temps de délai est calculé par la formule : $\textit{Délai}_j = x_{n+j} \times 1,5 \times d_{max}$, où $\textit{Délai}_j$ est le temps de délai de l'opération $j$ et $d_{max}$ est le plus grand temps de réalisation pour une opération dans l'instance. À chaque itération, l'opération avec la plus grande priorité est sélectionnée parmi celles dont l'opération précédente sur le lot est dans l'intervalle $[t, t+\textit{Délai}_j]$ (Ensemble $E$). Cette opération est réalisée le plus rapidement de par sa date de départ, son temps de réalisation et ses contraintes de précédence. La date de fin $t$ de l'ordonnancement est mise à jour à chaque itération. Cette procédure est répétée jusqu'à ce que toutes les opérations soient ordonnancées.

Les paramètres d'entrée de l'algorithme 3.1 sont : $g$ l'identifiant de l'itération, $S_g$ est l'ensemble des opérations déjà ordonnancées, $t_g$ la date associée à l'itération $g$ et $\textit{Délai}_g$ le temps de délai à l'itération $g$. D'autres ensembles ou paramètres sont utilisés dans l'algorithme 3.1 :

— $E_g$ : ensemble des opérations prêtes dans l'intervalle $[t, t+\text{Délai}_j]$.
— $tr_j$ : date de réalisation de l'opération $j$.
— $P_j$ : ensemble des opérations à réaliser avant l'opération $j$.
— $\textit{Priorité}_j$ : la priorité de l'opération $j$.
— $tr_j^l$ : date de fin de réalisation de l'opération précédente de $j$ sur le lot.
— $tr_j^m$ : date de fin de réalisation de l'opération précédente de $j$ sur la machine.
— $d_j$ : temps de réalisation de l'opération $j$.

### 3.3.5 *Multi-Objective Artificial Bee Colony* (MOABC)

Pour résoudre le FJSP multiobjectif par un MOABC, le *Pareto-based Discrete Artificial Bee Colony* (P-DABC) (Li et al., 2011a) est utilisé. Le PDABC reprend la trame générale de l'ABC avec trois phases, la *employed bee*, la *onlooker bee* et la *scoot bee*. Les deux premières phases utilisent différentes combinaisons de LS pour faire converger l'algorithme. La troisième phase diversifie la recherche de l'algorithme en créant de nouvelles solutions que pour en remplacer dans la population. Le PDABC possède une archive pour stocker les solutions non-dominées.

La source de nourriture est représentée par deux vecteurs (Tay and Wibowo, 2004) : le vecteur d'ordonnancement et le vecteur d'affectation. Le vecteur d'ordonnancement décrit la séquence des opérations sur les machines. Le vecteur d'affectation contient l'affectation des opérations aux machines.

Pour obtenir un voisinage de bonne qualité et diversifié, deux types de recherche locale sont appliqués pour le P-DABC.

**Algorithme 3.1** Algorithme de construction de l'ordonnancement

**Paramètre** $g = 0$

**Paramètre** $S_g = \{\oslash\}$

**Paramètre** $t_g = 0$

**Paramètre** $\text{Délai}_g = \text{Délai}_1$

1: **Tant que** $|S_g| < n + 1$ **Faire**
2: $E_g = \{j \in O\ S_{g-1} | tr_i \leq t_g + Délai_g,\ i \in P_j\}$
3: **Tant que** $E_g \neq \oslash$ **Faire**
4: $j = \max_{j \in E_g} \{\text{Priorité}_j\}$
5: $tr_j = max\left\{tr_j^l, tr_j^m\right\} + d_j$
6: $S_{g+1} = S_g \cup \{O_j\}$
7: $E_{g+1} = E_g\ \{O_j\}$
8: $t_{g+1} = min\{t_g, tr_j\}$
9: **Si** $t_{g+1} = tr_j$ **Alors**
10: $\text{Délai}_{g+1} = \text{Délai}_j$
11: **Fin Si**
12: $g = g + 1$
13: **Fin Tant que**
14: **Fin Tant que**

Le voisinage dans le vecteur de l'ordonnancement est une perturbation dans l'ordre des gènes le composant. Deux opérateurs peuvent être utilisés :

1. L'opérateur d'insertion (ou *insert*) déplace un numéro d'une position $j$ à une position $k$, avec $j \neq k$.
2. L'opérateur échange (ou *swap*) (Pan et al., 2009) intervertit deux symboles à des positions différentes $j$ et $k$, tel que $j \neq k$.

Dans le vecteur d'affectation, le voisinage est défini par le changement aléatoire d'un numéro sur une position. Cet opérateur, s ?exécute en plusieurs étapes :

1. Sélectionner aléatoirement la position d'une opération $O_{i,j}$ qui peut être effectuée par au moins deux machines.
2. Choisir au hasard une machine $M_k$ venant de $M_{i,j}$ qui est différente de la machine utilisée à présent.
3. La position de $O_{i,j}$ prend la valeur de l'indice de la machine $M_k$.

Lors de la phase *employed bee*, une abeille effectue une recherche locale autour de la source de nourriture qui lui est attribuée. Autrement dit, l'*employed bee* fait une prospection avec les opérateurs de recherche locale définis précédemment. Après avoir réalisé une recherche locale, l'*employed bee* obtient une nouvelle source de nourriture voisine de l'initiale. Cette nouvelle source de nour-

riture est évaluée et comparée à l'ancienne. Comme dans le classique ABC, la meilleure source de nourriture est gardée. Si aucune des solutions ne domine, l'*employed bee* garde une solution choisie au hasard.

Dans l'ABC classique, chaque *employed bee* effectue une recherche locale pour trouver une meilleure source de nourriture. Une recherche locale sur une source de nourriture est effectuée en indépendance par rapport au reste de la population. Pour le FJSP, la partie affectation détient les informations pour l'attribution des opérations aux machines. Pour obtenir une meilleure combinaison d'affectation, une abeille peut aller chercher une partie de nouvelles informations sur une autre source de nourriture. Dans cet algorithme hybride, chaque *employed bee* sélectionne aléatoirement une autre source de nourriture, pour effectuer un croisement sur les vecteurs d'affectation des deux solutions. Un croisement à deux points est appliqué, il est similaire au croisement à un point (Pezzella et al., 2008). Deux points de troncature sont choisis aléatoirement, pour constituer trois parties, la deuxième étant échangée avec celle de l'autre source de nourriture. Comme pour la recherche locale, l'ancienne et la nouvelle solution sont comparées. La meilleure source de nourriture est gardée, si aucune ne domine, le choix est fait aléatoirement.

Lors de la phase *onlooker bee*, une abeille va sélectionner une source de nourriture de qualité. Pour le FJSP, le P-DABC utilise un système de tournoi pour sélectionner cette source de nourriture. Pour la sélection par tournoi, trois sources de nourriture sont prises au hasard dans la population. La source qui domine les deux autres est retenue par l'*onlooker bee*. S'il existe au moins deux solutions non-dominées alors la source de nourriture gardée est choisie au hasard. L'*onlooker bee* exécute sur les sources de nourriture le même type de recherche locale qu'une *employed bee*. Elle explore le voisinage avec les opérateurs définis précédemment et trouve une nouvelle source de nourriture. La meilleure entre l'ancienne et la nouvelle source de nourriture est mémorisée dans la population.

Une *scoot bee* recherche aléatoirement dans l'espace de solution de nouvelles sources de nourriture. Ceci permet de diversifier la population de sources de nourriture et d'éviter les minimums locaux. Pour une diversification de qualité, des solutions du front de Pareto sont sélectionnées aléatoirement par les *scoot bee*. Ces solutions non-dominées sont dans des régions prometteuses. Une nouvelle solution est construite par une série d'opérateurs d'échange dans le vecteur ordonnancement et de changement dans le vecteur d'affectation sur une solution non-dominée. Ainsi, la nouvelle source de nourriture est différente du reste de la population tout en étant proche du front de Pareto optimal.

Les paramètres du PDABC sont la taille de la population, les nombres d' *employed bee*, d'*onlooker bee* et de *scoot bee*. Le PDABC suit la trame générale suivante :

**Étape 1 :** Initialisation de la population, évaluation et sauvegarde des solutions non-dominées dans l'archive.

**Étape 2 :** Phase *employed bee*.

**Étape 2.1 :** Placer une *employed bee* sur chaque source de nourriture.

**Étape 2.2 :** Explorer le voisinage de chaque source de nourriture pour trouver une nouvelle source de nourriture et garder la meilleure des deux solutions.

**Étape 2.3 :** Croiser le vecteur affectation de chaque *employed bee* avec celui d'une autre et garder la meilleure entre l'ancienne et la nouvelle solution.

**Étape 3 :** Phase *onlooker bee*.

**Étape 3.1 :** Chaque *onlooker bee* choisit une source de nourriture par un système de tournoi à trois participants.

**Étape 3.2 :** Explorer le voisinage pour trouver une nouvelle source de nourriture et garder la meilleure des deux solutions.

**Étape 3.3 :** Classer les sources de nourriture par rang de Pareto et sauvegarder les solutions non-dominées dans l'archive.

**Étape 4 :** Phase *scoot bee*.

**Étape 4.1 :** Chaque *scoot bee* choisit aléatoirement une source de nourriture contenue dans l'archive.

**Étape 4.2 :** Explorer le voisinage aléatoirement pour générer une nouvelle solution.

**Étape 4.2 :** Remplacer une solution de la population par cette nouvelle source de nourriture.

**Étape 5 :** Arrêt si le critère est atteint sinon retourner à l'étape 2.

### 3.3.6 Ajustement des opérateurs de la MCE pour l'optimisation multiobjectif

La MCE garde la même procédure pour résoudre les problèmes d'optimisation mono et multiobjectif. Mais les opérateurs d'évaluation de la MCE, des tests de convergence et de recherche locale sont adaptés à l'optimisation multiobjectif.

Pour l'opérateur d'orientation (section 2.7), les meilleures solutions dans la population sont remplacées par les solutions non-dominées. Les zones *nd* contiennent donc au moins une solution non-dominée de la population.

L'indicateur MCE (section 2.7.2), au lieu de comparer la note obtenue à la fonction objectif par l'ancienne et la nouvelle solution, l'ancien et le nouveau fronts de Pareto sont évalués avec la métrique C (voir section 3.4.3). Si la progression du front de Pareto est faible, une phase de recherche locale est effectuée.

Pour la recherche locale, une solution voisine n'est sélectionnée que si elle domine strictement au sens de Pareto l'ancienne solution, c'est-à-dire que la nouvelle solution domine l'ancienne au moins sur tous les objectifs.

## 3.4 Expérimentations

La présentation des expérimentations est divisée en deux parties. La première énumère les instances, la population initiale, les métriques et les paramètres pour les expérimentations. Les instances sont issues de la littérature et créées par un générateur d'instances pour les plus grandes tailles. La procédure de génération de la population initiale est identique pour le FJSP mono-objectif et multiobjectif. Les paramètres sont choisis après une série de plusieurs tests.

La deuxième partie pour les expérimentations est aussi séparée en trois phases. La première phase, le NSGAII est hybridé avec plusieurs variantes de la MCE avec la recherche locale. Le comportement de ces différentes hybridations est observé en comptabilisant le nombre de PRL et d'application de LS sur les solutions non-dominées, dominées et créées dans les zones inexplorées. Dans la deuxième phase, le NSGAII et les hybridations sont comparés par la métrique C. L'objectif est de trouver la meilleure hybridation parmi celles proposées. En dernier, la meilleure hybridation est adaptée à d'autres algorithmes multiobjectifs : SPEA2, MOAIS, MOACO, MOPSO et MOABC. Dans un premier temps, les algorithmes classiques sont comparés par la métrique C entre eux pour déterminer le meilleur. Ensuite, les algorithmes classiques sont comparés à leur hybridation par la MCE pour en connaître l'influence sur chacun. En dernier, les algorithmes hybridés sont confrontés entre eux pour trouver le plus performant.

Les tests sont faits sur un ordinateur avec un processeur core i5 3.2 GHz et 8 Giga de RAM sous le système d'exploitation Windows 7. Le langage utilisé est le C++ sous la plate-forme IDE codeblock.

### 3.4.1 Générateur d'instances

Les algorithmes sont testés sur des instances rencontrées fréquemment dans la littérature (Kacem et al., 2002b), (Kacem et al., 2002a) et (Xia and Wu, 2005) et des instances générées aléatoirement (Xing et al., 2010). Pour rappel de la section 3.3.1, les instances du FJSP sont caractérisées par $(n/h/m)$ avec $n$ le nombre de lots, $h$ la taille maximale de la gamme opératoire d'un lot et $m$ le nombre de machines.

Une instance est composée de deux ensembles : les lots $J$ et les machines $M$. Chaque lot $J_{1\leq i\leq n}$ est composé d'une suite ordonnée d'opérations. Une opération $O_{ij}$ peut être réalisée par un ensemble de machines $M_{ij} \subset M$.

Les tableaux 3.5, 3.6, 3.7 et 3.8 présentent les instances utilisées par (Kacem et al., 2002b), (Kacem et al., 2002a) et (Xia and Wu, 2005). Les lignes décrivent les lots et leur gamme opératoire et les colonnes les machines. L'intersection entre une colonne et une ligne donne le temps de réalisation de l'opération $O_{i,j}$ sur la machine $M_k$. Note si la case contient le symbole $-$, l'opération $O_{i,j}$ n'est pas réalisable sur la machine $M_k$. La dernière colonne contient les dates d'échéance $d_i$ des lots calculer par l'équation 3.3 avec comme coefficient $cd = 1,5$.

TABLEAU 3.5: Problème de 10 lots et 7 machines avec 29 opérations (Kacem et al., 2002b)

| Instance $I_1$ (10/3/7) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ | $M_6$ | $M_7$ | $d_i$ |
| $J_1$ | $O_{1,1}$ | 1 | 4 | 6 | 9 | 3 | 5 | 2 | |
| | $O_{1,2}$ | 8 | 9 | 5 | 4 | 1 | 1 | 3 | |
| | $O_{1,3}$ | 4 | 8 | 10 | 4 | 11 | 4 | 3 | 23 |
| $J_2$ | $O_{2,1}$ | 6 | 9 | 8 | 6 | 5 | 10 | 3 | |
| | $O_{2,2}$ | 2 | 10 | 4 | 5 | 9 | 8 | 4 | 20 |
| $J_3$ | $O_{3,1}$ | 15 | 4 | 8 | 4 | 8 | 7 | 1 | |
| | $O_{3,2}$ | 9 | 6 | 1 | 10 | 7 | 1 | 6 | |
| | $O_{3,3}$ | 11 | 2 | 7 | 5 | 2 | 3 | 14 | 29 |
| $J_4$ | $O_{4,1}$ | 2 | 8 | 5 | 8 | 9 | 4 | 3 | |
| | $O_{4,2}$ | 5 | 3 | 8 | 1 | 9 | 3 | 6 | |
| | $O_{4,3}$ | 1 | 2 | 6 | 4 | 1 | 7 | 2 | 21 |
| $J_5$ | $O_{5,1}$ | 7 | 1 | 8 | 5 | 4 | 3 | 9 | |
| | $O_{5,2}$ | 2 | 4 | 5 | 10 | 6 | 4 | 9 | |
| | $O_{5,3}$ | 5 | 1 | 7 | 1 | 6 | 6 | 2 | 23 |
| $J_6$ | $O_{6,1}$ | 8 | 7 | 4 | 56 | 9 | 8 | 4 | |
| | $O_{6,2}$ | 5 | 14 | 1 | 9 | 6 | 5 | 8 | |
| | $O_{6,3}$ | 3 | 5 | 2 | 5 | 4 | 5 | 7 | 38 |
| $J_7$ | $O_{7,1}$ | 5 | 6 | 3 | 6 | 5 | 15 | 2 | |
| | $O_{7,2}$ | 6 | 5 | 4 | 9 | 5 | 4 | 3 | |
| | $O_{7,3}$ | 9 | 8 | 2 | 8 | 6 | 1 | 7 | 26 |
| $J_8$ | $O_{8,1}$ | 6 | 1 | 4 | 1 | 10 | 4 | 3 | |
| | $O_{8,2}$ | 11 | 13 | 9 | 8 | 9 | 10 | 8 | |
| | $O_{8,3}$ | 4 | 2 | 7 | 8 | 3 | 10 | 7 | 30 |
| $J_9$ | $O_{9,1}$ | 12 | 5 | 4 | 8 | 4 | 5 | 5 | |
| | $O_{9,2}$ | 4 | 2 | 15 | 99 | 4 | 7 | 3 | |
| | $O_{9,3}$ | 9 | 5 | 11 | 2 | 5 | 4 | 2 | 46 |
| $J_{10}$ | $O_{10,1}$ | 9 | 4 | 13 | 10 | 7 | 6 | 8 | |
| | $O_{10,2}$ | 4 | 3 | 25 | 3 | 8 | 1 | 2 | |
| | $O_{10,3}$ | 1 | 2 | 6 | 11 | 13 | 3 | 5 | 31 |

TABLEAU 3.6: Problème de 10 lots et 10 machines avec 27 opérations (Kacem et al., 2002a)

| Instance $I_2$ (10/4/10) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ | $M_6$ | $M_7$ | $M_8$ | $d_i$ |
| $J_1$ | $O_{1,1}$ | 5 | 3 | 5 | 3 | 3 | - | 10 | 9 | |
| | $O_{1,2}$ | 10 | - | 5 | 8 | 3 | 9 | 9 | 6 | |
| | $O_{1,3}$ | - | 10 | - | 5 | 6 | 2 | 4 | 5 | 27 |
| $J_2$ | $O_{2,1}$ | 5 | 7 | 3 | 9 | 8 | - | 9 | - | |
| | $O_{2,2}$ | - | 8 | 5 | 2 | 6 | 7 | 10 | 9 | |
| | $O_{2,3}$ | - | 10 | - | 5 | 6 | 4 | 1 | 7 | |
| | $O_{2,4}$ | 10 | 8 | 9 | 6 | 4 | 7 | - | - | 40 |
| $J_3$ | $O_{3,1}$ | 10 | - | - | 7 | 6 | 5 | 2 | 4 | |
| | $O_{3,2}$ | - | 10 | 6 | 4 | 8 | 9 | 10 | - | |
| | $O_{3,3}$ | 1 | 4 | 5 | 6 | - | 10 | - | 7 | 29 |
| $J_4$ | $O_{4,1}$ | 3 | 1 | 6 | 5 | 9 | 7 | 8 | 4 | |
| | $O_{4,2}$ | 12 | 11 | 7 | 8 | 10 | 5 | 6 | 9 | |
| | $O_{4,3}$ | 4 | 6 | 2 | 10 | 3 | 9 | 5 | 7 | 30 |
| $J_5$ | $O_{5,1}$ | 3 | 6 | 7 | 8 | 9 | - | 10 | - | |
| | $O_{5,2}$ | 10 | - | 7 | 4 | 9 | 8 | 6 | - | |
| | $O_{5,3}$ | - | 9 | 8 | 7 | 4 | 2 | 7 | - | |
| | $O_{5,4}$ | 11 | 9 | - | 6 | 7 | 5 | 3 | 6 | 42 |
| $J_6$ | $O_{6,1}$ | 6 | 7 | 1 | 4 | 6 | 9 | - | 10 | |
| | $O_{6,2}$ | 11 | - | 9 | 9 | 9 | 7 | 6 | 4 | |
| | $O_{6,3}$ | 10 | 5 | 9 | 10 | 11 | - | 10 | - | 35 |
| $J_7$ | $O_{7,1}$ | 5 | 4 | 2 | 6 | 7 | - | 10 | - | |
| | $O_{7,2}$ | - | 9 | - | 9 | 11 | 9 | 10 | 5 | |
| | $O_{7,3}$ | - | 8 | 9 | 3 | 8 | 6 | - | 10 | 33 |
| $J_8$ | $O_{8,1}$ | 2 | 8 | 5 | 9 | - | 4 | - | 10 | |
| | $O_{8,2}$ | 7 | 4 | 7 | 8 | 9 | - | 10 | - | |
| | $O_{8,3}$ | 9 | 9 | - | 8 | 5 | 6 | 7 | 1 | |
| | $O_{8,4}$ | 9 | - | 3 | 7 | 1 | 5 | 8 | - | 39 |

TABLEAU 3.7: Problème de 10 lots et 10 machines avec 30 opérations (Kacem et al., 2002a)

| Instance $I_3$ (10/3/10) | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ | $M_6$ | $M_7$ | $M_8$ | $M_9$ | $M_{10}$ | $d_i$ |
| $J_1$ | $O_{1,1}$ | 1 | 4 | 6 | 9 | 3 | 5 | 2 | 8 | 9 | 5 | |
| | $O_{1,2}$ | 4 | 1 | 1 | 3 | 4 | 8 | 10 | 4 | 11 | 4 | |
| | $O_{1,3}$ | 3 | 2 | 5 | 1 | 5 | 6 | 9 | 5 | 10 | 3 | 23 |
| $J_2$ | $O_{2,1}$ | 2 | 10 | 4 | 5 | 9 | 8 | 4 | 15 | 8 | 4 | |
| | $O_{2,2}$ | 4 | 8 | 7 | 1 | 9 | 6 | 1 | 10 | 7 | 1 | |
| | $O_{2,3}$ | 6 | 11 | 2 | 7 | 5 | 3 | 5 | 3 | 8 | 1 | 29 |
| $J_3$ | $O_{3,1}$ | 8 | 5 | 8 | 9 | 4 | 3 | 5 | 3 | 8 | 1 | |
| | $O_{3,2}$ | 9 | 3 | 6 | 1 | 2 | 6 | 4 | 1 | 7 | 2 | |
| | $O_{3,3}$ | 7 | 1 | 8 | 5 | 4 | 9 | 1 | 2 | 3 | 4 | 21 |
| $J_4$ | $O_{4,1}$ | 5 | 10 | 6 | 4 | 9 | 5 | 1 | 7 | 1 | 6 | |
| | $O_{4,2}$ | 4 | 2 | 3 | 8 | 7 | 4 | 6 | 9 | 8 | 4 | |
| | $O_{4,3}$ | 7 | 3 | 12 | 1 | 6 | 5 | 8 | 3 | 5 | 2 | 25 |
| $J_5$ | $O_{5,1}$ | 7 | 10 | 4 | 5 | 6 | 3 | 5 | 15 | 2 | 6 | |
| | $O_{5,2}$ | 5 | 6 | 3 | 9 | 8 | 2 | 8 | 6 | 1 | 7 | |
| | $O_{5,3}$ | 6 | 1 | 4 | 1 | 10 | 4 | 3 | 11 | 13 | 9 | 27 |
| $J_6$ | $O_{6,1}$ | 8 | 9 | 10 | 8 | 4 | 2 | 7 | 8 | 3 | 10 | |
| | $O_{6,2}$ | 7 | 3 | 12 | 5 | 4 | 3 | 6 | 9 | 2 | 15 | |
| | $O_{6,3}$ | 4 | 7 | 3 | 6 | 3 | 4 | 1 | 5 | 1 | 11 | 27 |
| $J_7$ | $O_{7,1}$ | 1 | 7 | 8 | 3 | 4 | 9 | 4 | 13 | 10 | 7 | |
| | $O_{7,2}$ | 3 | 8 | 1 | 2 | 3 | 6 | 11 | 2 | 13 | 3 | |
| | $O_{7,3}$ | 5 | 4 | 2 | 1 | 2 | 1 | 8 | 14 | 5 | 7 | 26 |
| $J_8$ | $O_{8,1}$ | 5 | 7 | 11 | 3 | 2 | 9 | 8 | 5 | 12 | 8 | |
| | $O_{8,2}$ | 8 | 3 | 10 | 7 | 5 | 13 | 4 | 6 | 8 | 4 | |
| | $O_{8,3}$ | 6 | 2 | 13 | 5 | 4 | 3 | 5 | 7 | 9 | 5 | 30 |
| $J_9$ | $O_{9,1}$ | 3 | 9 | 1 | 3 | 8 | 1 | 6 | 7 | 5 | 4 | |
| | $O_{9,2}$ | 4 | 6 | 2 | 5 | 7 | 3 | 1 | 9 | 6 | 7 | |
| | $O_{9,3}$ | 8 | 5 | 4 | 8 | 6 | 1 | 2 | 3 | 10 | 12 | 24 |
| $J_{10}$ | $O_{10,1}$ | 4 | 3 | 1 | 6 | 7 | 1 | 2 | 6 | 20 | 6 | |
| | $O_{10,2}$ | 3 | 1 | 8 | 1 | 9 | 4 | 1 | 4 | 17 | 15 | |
| | $O_{10,3}$ | 9 | 2 | 4 | 2 | 3 | 5 | 2 | 4 | 10 | 23 | 28 |

TABLEAU 3.8: Problème de 15 lots et 10 machines avec 56 opérations (Xia and Wu, 2005)

| Instance $I_4$ (15/4/10) | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ | $M_6$ | $M_7$ | $M_8$ | $M_9$ | $M_{10}$ | $d_i$ |
| $J_1$ | $O_{1,1}$ | 1 | 4 | 6 | 9 | 3 | 5 | 2 | 8 | 9 | 4 | |
| | $O_{1,2}$ | 1 | 1 | 3 | 4 | 8 | 10 | 4 | 11 | 4 | 3 | |
| | $O_{1,3}$ | 2 | 5 | 1 | 5 | 6 | 9 | 5 | 10 | 3 | 2 | |
| | $O_{1,4}$ | 10 | 4 | 5 | 9 | 8 | 4 | 15 | 8 | 4 | 4 | 33 |
| $J_2$ | $O_{2,1}$ | 4 | 8 | 7 | 1 | 9 | 6 | 1 | 10 | 7 | 1 | |
| | $O_{2,2}$ | 6 | 11 | 2 | 7 | 5 | 3 | 5 | 14 | 9 | 2 | |
| | $O_{2,3}$ | 8 | 5 | 8 | 9 | 4 | 3 | 5 | 3 | 8 | 1 | |
| | $O_{2,4}$ | 9 | 3 | 6 | 1 | 2 | 6 | 4 | 1 | 7 | 2 | 32 |
| $J_3$ | $O_{3,1}$ | 7 | 1 | 8 | 5 | 4 | 9 | 1 | 2 | 3 | 4 | |
| | $O_{3,2}$ | 5 | 10 | 6 | 4 | 9 | 5 | 1 | 7 | 1 | 6 | |
| | $O_{3,3}$ | 4 | 2 | 3 | 8 | 7 | 4 | 6 | 9 | 8 | 4 | |
| | $O_{3,4}$ | 7 | 3 | 12 | 1 | 6 | 5 | 8 | 3 | 5 | 2 | 31 |
| $J_4$ | $O_{4,1}$ | 6 | 2 | 5 | 4 | 1 | 2 | 3 | 6 | 5 | 4 | |
| | $O_{4,2}$ | 8 | 5 | 7 | 4 | 1 | 2 | 36 | 5 | 8 | 5 | |
| | $O_{4,3}$ | 9 | 6 | 2 | 4 | 5 | 1 | 3 | 6 | 5 | 2 | |
| | $O_{4,4}$ | 11 | 4 | 5 | 6 | 2 | 7 | 5 | 4 | 2 | 1 | 32 |
| $J_5$ | $O_{5,1}$ | 6 | 9 | 2 | 3 | 5 | 8 | 7 | 4 | 1 | 2 | |
| | $O_{5,2}$ | 5 | 4 | 6 | 3 | 5 | 2 | 28 | 7 | 4 | 5 | |
| | $O_{5,3}$ | 6 | 2 | 4 | 3 | 6 | 5 | 2 | 4 | 7 | 9 | |
| | $O_{5,4}$ | 6 | 5 | 4 | 2 | 3 | 2 | 5 | 4 | 7 | 5 | 31 |
| $J_6$ | $O_{6,1}$ | 4 | 1 | 3 | 2 | 6 | 9 | 8 | 5 | 4 | 2 | |
| | $O_{6,2}$ | 1 | 3 | 6 | 5 | 4 | 7 | 5 | 4 | 6 | 5 | 14 |
| $J_7$ | $O_{7,1}$ | 1 | 4 | 2 | 5 | 3 | 6 | 9 | 8 | 5 | 4 | |
| | $O_{7,2}$ | 2 | 1 | 4 | 5 | 2 | 3 | 5 | 4 | 2 | 5 | 12 |
| $J_8$ | $O_{8,1}$ | 2 | 3 | 6 | 2 | 5 | 4 | 1 | 5 | 8 | 7 | |
| | $O_{8,2}$ | 4 | 5 | 6 | 2 | 3 | 5 | 4 | 1 | 2 | 5 | |
| | $O_{8,3}$ | 3 | 5 | 4 | 2 | 5 | 49 | 8 | 5 | 4 | 5 | |
| | $O_{8,4}$ | 1 | 2 | 36 | 5 | 2 | 3 | 6 | 4 | 11 | 2 | 37 |
| $J_9$ | $O_{9,1}$ | 6 | 3 | 2 | 22 | 44 | 11 | 10 | 23 | 5 | 1 | |
| | $O_{9,2}$ | 2 | 3 | 2 | 12 | 15 | 10 | 12 | 14 | 18 | 16 | |
| | $O_{9,3}$ | 20 | 17 | 12 | 5 | 9 | 6 | 4 | 7 | 5 | 6 | |
| | $O_{9,4}$ | 9 | 8 | 7 | 4 | 5 | 8 | 7 | 4 | 56 | 2 | 65 |

suite à la page suivante

| Instance $I_4$ (15/4/10) (suite) | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ | $M_6$ | $M_7$ | $M_8$ | $M_9$ | $M_{10}$ | $d_i$ |
| $J_{10}$ | $O_{10,1}$ | 5 | 8 | 7 | 4 | 56 | 3 | 2 | 5 | 4 | 1 | |
| | $O_{10,2}$ | 2 | 5 | 6 | 9 | 8 | 5 | 4 | 2 | 5 | 4 | |
| | $O_{10,3}$ | 6 | 3 | 2 | 5 | 4 | 7 | 4 | 5 | 2 | 1 | |
| | $O_{10,4}$ | 3 | 2 | 5 | 6 | 5 | 8 | 7 | 4 | 5 | 2 | 35 |
| $J_{11}$ | $O_{11,1}$ | 1 | 2 | 3 | 6 | 5 | 2 | 1 | 4 | 2 | 1 | |
| | $O_{11,2}$ | 2 | 3 | 6 | 3 | 2 | 1 | 4 | 10 | 12 | 1 | |
| | $O_{11,3}$ | 3 | 6 | 2 | 5 | 8 | 4 | 6 | 3 | 2 | 5 | |
| | $O_{11,4}$ | 4 | 1 | 45 | 6 | 2 | 4 | 1 | 25 | 2 | 4 | 32 |
| $J_{12}$ | $O_{12,1}$ | 9 | 8 | 5 | 6 | 3 | 6 | 5 | 2 | 4 | 2 | |
| | $O_{12,2}$ | 5 | 8 | 9 | 5 | 4 | 75 | 63 | 6 | 5 | 21 | |
| | $O_{12,3}$ | 12 | 5 | 4 | 6 | 3 | 2 | 5 | 4 | 2 | 5 | |
| | $O_{12,4}$ | 8 | 7 | 9 | 5 | 6 | 3 | 2 | 5 | 8 | 4 | 54 |
| $J_{13}$ | $O_{13,1}$ | 4 | 2 | 5 | 6 | 8 | 5 | 6 | 4 | 6 | 2 | |
| | $O_{13,2}$ | 3 | 5 | 4 | 7 | 5 | 8 | 6 | 6 | 3 | 2 | |
| | $O_{13,3}$ | 5 | 4 | 5 | 8 | 5 | 4 | 6 | 5 | 4 | 2 | |
| | $O_{13,4}$ | 3 | 2 | 5 | 6 | 5 | 4 | 8 | 5 | 6 | 4 | 29 |
| $J_{14}$ | $O_{14,1}$ | 2 | 3 | 5 | 4 | 6 | 5 | 4 | 85 | 4 | 5 | |
| | $O_{14,2}$ | 6 | 2 | 4 | 5 | 8 | 6 | 5 | 4 | 2 | 6 | |
| | $O_{14,3}$ | 3 | 25 | 4 | 8 | 5 | 6 | 3 | 2 | 5 | 4 | |
| | $O_{14,4}$ | 8 | 5 | 6 | 4 | 2 | 3 | 6 | 8 | 5 | 4 | 44 |
| $J_{15}$ | $O_{15,1}$ | 2 | 5 | 6 | 8 | 5 | 6 | 3 | 2 | 5 | 4 | |
| | $O_{15,2}$ | 5 | 6 | 2 | 5 | 4 | 2 | 5 | 3 | 2 | 5 | |
| | $O_{15,3}$ | 4 | 5 | 2 | 3 | 5 | 2 | 8 | 4 | 7 | 5 | |
| | $O_{15,4}$ | 6 | 2 | 11 | 14 | 2 | 3 | 6 | 5 | 4 | 8 | 29 |

Pour obtenir des instances de tailles plus variées, un générateur est adapté (Xing et al., 2010) pour les objectifs de minimisation et de production juste à temps. Il prend comme paramètres le nombre de lots, de machines et la taille de la gamme opératoire des lots. Le temps de réalisation d'une opération d'une machine à l'autre est encadré par une fenêtre définissant la variation de temps d'exécution d'une machine à l'autre. La date d'échéance des lots est ajoutée au générateur et calculée par l'équation 3.3.

Les paramètres pris en compte sont les suivants :

— $n$ : Le nombre de lots.

— $m$ : Le nombre de machines.

— $l$ et $h$ : Les nombres minimum et maximum d'opérations dans la gamme opératoire des lots.

— $tr_{min}$ et $tr_{max}$ : La fourchette de temps de réalisation $p_{ij}$ pour une opération $O_{ij}$.

— $tpv_{min}$ et $tpv_{max}$ : La déviation admise par rapport à $p_{ij}$ d'une machine à l'autre.
— $cd$ : Le coefficient pour calculer la date d'échéance de chaque lot (voir équation 3.3).

### 3.4.2 Population initiale

La population initiale utilisée par les algorithmes multiobjectifs est identique à celle par les algorithmes mono-objectifs.

Pour rappel, l'initialisation de l'affectation des opérations aux machines se fait avec les règles du minimum global et d'affectation par permutations aléatoires (Kacem et al., 2002a). Pour déterminer l'ordonnancement des opérations sont utilisées une règle séquençant les opérations aléatoirement (*RANDOM*), la règle du *Most Work Remaining* (MWR) et celle du *Most Operation Rule* (MOR).

Chacune de ces règles est utilisée pour créer une partie de la population garantissant une bonne répartition de la population dans l'espace des solutions. Les règles du minimum global et du MWR obtiennent des résultats de bonne qualité. Les règles par permutations aléatoires, RANDOM et MOR assurent la diversité des gènes dans la population.

### 3.4.3 Métriques

Les métriques utilisées pour évaluer les performances sont la métrique C et la métrique H pour la convergence, la métrique de diversification, le nombre de solutions non-dominées fourni par l'algorithme et le temps d'exécution de l'algorithme.

Compter le nombre de solutions non-dominées est le moyen le plus simple pour mesurer la diversité de l'approximation d'un front de Pareto optimal. Pour cette énumération, les solutions dont les objectifs sont égaux ne sont comptées que pour une solution.

La métrique C est utilisée pour comparer les approximations des fronts de Pareto optimaux, notées $F_1$ et $F_2$, fournis par deux algorithmes. Un nombre réel entre [0,1] représente la comparaison d'un front $F_1$ par rapport à un autre $F_2$ (voir équation 3.14). La dominance stricte est utilisée pour comparer deux solutions, cela signifie qu'une solution $x_1$ domine une solution $x_2$ quand $x_1$ obtient une meilleure note que $x_2$ sur un objectif et des notes au moins égales à $x_2$ sur les autres objectifs.

$$C(F_1, F_2) = \frac{|\{\vec{x_2} \in F_2; \exists \vec{x_1} \in F_1 | \vec{x_1} \prec \vec{x_2}\}|}{|F_2|} \tag{3.14}$$

Où $|F|$ est le cardinal de l'ensemble $F$, c'est-à-dire le nombre de solutions dans $F$ et $\vec{x_1} \prec \vec{x_2}$ signifie que la solution $x_1$ domine strictement la solution $x_2$.

$C(F_1, F_2)$ renvoie un réel dans l'intervalle [0,1] représentant le nombre solution de $F_2$ dominées par au moins une solution de $F_1$. Remarque, $C(F_1, F_2) \neq 1 - C(F_2, F_1)$ donc pour comparer deux fronts de Pareto, il est nécessaire de calculer $C(F_1, F_2)$ et $C(F_2, F_1)$.

La métrique H ou hypersurface est utilisée pour mesurer la surface couverte entre les solutions d'un front de Pareto $F$ et les axes des objectifs.

$$H = \left\{ \bigcup_i a_i \| x_i \in F \right\} \tag{3.15}$$

avec $a_i$ la surface de la solution $x_i$.

Pour comparer les hypersurfaces de deux fronts de Pareto $F_1$ et $F_2$, le rapport est fait entre les deux surfaces (voir équation 3.15).

$$HR = \frac{H_1}{H_2} \tag{3.16}$$

$H_1$ et $H_2$ sont les surfaces occupées par les solutions de $F_1$ et $F_2$.

La métrique de Diversification (DM) évalue la diversité des solutions non-dominées retournées par un algorithme. Elle évalue la distance entre les points extrêmes d'un front de Pareto. Pour cette métrique, un algorithme doit obtenir la valeur la plus haute possible. L'équation 3.17 est utilisée pour la calculer.

$$DM = \sqrt{\sum_j^m (max\{f_{ji}\} - min\{f_{ji}\})^2} \tag{3.17}$$

avec $m$ le nombre d'objectifs considérés et $f_{ji}$ l'ensemble des notes obtenues par les solutions non-dominées pour l'objectif $j$.

### 3.4.4 Paramétrages

Les paramètres sont déterminés empiriquement après plusieurs tests, les paramètres gardés sont ceux qui offrent les meilleurs résultats en terme de performance et de stabilité.

Les instances générées sont des tailles suivantes : ($I_5$ 10/5/10), ($I_6$ 10/5/20), ($I_7$ 10/5/30), ($I_8$ 15/5/15), ($I_9$ 15/5/30) et ($I_{10}$ 15/5/45). Les autres paramètres pour la génération des instances sont :

— $l = 1$.
— $h = 5$.
— $pt_{min} = 5$.
— $pt_{max} = 15$.
— $cd = 1,5$.

Les individus de la population initiale sont générés par les heuristiques dans les proportions suivantes :

— $pmg = 20\%$ pour la règle du minimum global.

— $papa = 80\%$ pour la règle de permutation aléatoire.

— $pr = 20\%$ par l'heuristique d'ordonnancement aléatoire.

— $pmw = 40\%$ par la méthode MWR.

— $pmo = 40\%$ par la méthode MOR.

Les paramètres communs à tous les algorithmes sont :

— taille de la population : 100.

— nombre d'itérations : 2000.

— nombre de simulation : 10.

Les paramètres retenus pour le NSGAII et le SPEA2 sont les suivants :

— probabilité pour le croisement : 90%.

— probabilité pour la mutation : $10/n\%$ avec n le nombre d'operations dans l'instance.

Les paramètres sélectionnés pour le MOACO sont les suivants :

— les taux minimum et maximum dans les mémoires de phéromones : $\tau_{min} = 0,01$ et $\tau_{max} = 1,00$.

— le taux d'évaporation de phéromones : $\rho = 0,025$.

— le taux de dépôt de phéromones pour OAMK et OAPK : $Q_1 = 5$ et $Q_2 = 0,125$.

— les coefficients de pondération de OAPK : $\alpha = 5$ et $\beta = 5$.

Les paramètres utilisés pour le MOPSO sont :

— $w_{min} = 0,4$ et $w_{max} = 0,9$ : le poids inertiel minimum et maximum.

— $C_1 = 0,7$ et $C_2 = 1,3$ : les coefficients des minimums global et local.

Les paramètres du MOABC sont :

— le nombre d' *employed bee* : 100.

— le nombre d' *onlooker bee* : 100.

— le nombre de *scoot bee* : 20.

Les instances générées sont des tailles suivantes : ($I_5$ 10/5/10), ($I_6$ 10/5/20), ($I_7$ 10/5/30), ($I_8$ 15/5/15), ($I_9$ 15/5/30) et ($I_{10}$ 15/5/45).

Les abréviations pour les diverses variantes d'hybridation avec la MCE sont :

— Alg-LS : l'algorithme Alg hybridé avec la recherche locale sur les solutions non-dominées.

— Alg-MCE1 : Alg hybridé avec la recherche locale sur les solutions dans les zones $znd$.

— Alg-MCE2 : Alg hybridé avec la recherche locale sur les solutions dans les zones $zfe$.

— Alg-MCE3 : Alg hybridé avec la recherche locale sur les solutions créées dans les zones $zi$.

— Alg-MCE4 : Alg avec la recherche locale sur les solutions dans les zones $znd$ et $zi$.

— Alg-MCE5 : Alg avec la recherche locale sur les solutions dans les zones $zfe$ et $zi$.

Les paramètres concernant la PRL des hybridations LS, Mce1, Mce2, Mce3, Mce4 et Mce5 sont :

— période d'application d'une PRL : $T = 100$.

— nombre d'essais avant arrêt d'une LS : 500.

— indicateur d'application : $Ind - Mce = 5\%$.

— nombre d'essais pour la création de solutions dans les $zi$ : 1000.

Les résultats présentés dans les tableaux sont les moyennes de 10 simulations.

### 3.4.5 Comportement des hybridations

Plusieurs hybridations sont étudiées dans cette partie, l'objectif est de comprendre le comportement de chacune avec comme critères le nombre de PRL et le nombre de LS appliquées sur des solutions non-dominées, dominées et créées dans les zones inexplorées et le temps d'exécution. Le très connu NSGAII est choisi comme algorithme de référence. Six variantes du NSGAII sont expérimentées : NSGAII, NSGAII-LS, NSGAII-MCE1, NSGAII-MCE2, NSGAII-MCE3, NSGAII-MCE4 et NSGAII-MCE5.

TABLEAU 3.9 – Nombre moyen de Phases de Recherches Locales (PRL)

| Instance | NSGAII - | | | | | |
|---|---|---|---|---|---|---|
| taille | LS | MCE1 | MCE2 | MCE3 | MCE4 | MCE5 |
| 7/3/10 | 20,0 | 13,1 | 13,4 | 13,4 | 13,5 | 13,3 |
| 8/4/8 | 20,0 | 13,5 | 14,9 | 13,6 | 14,1 | 10,6 |
| 10/3/10 | 20,0 | 14,8 | 14,9 | 14,6 | 14,8 | 14,6 |
| 10/4/15 | 20,0 | 12,1 | 12,3 | 17,8 | 10,6 | 10,3 |
| 10/5/10 | 20,0 | 14,3 | 14,5 | 11,5 | 10,9 | 12,6 |
| 10/5/20 | 20,0 | 11,4 | 10,1 | 8,9 | 9,9 | 10,0 |
| 10/5/30 | 20,0 | 10,4 | 9,6 | 9,0 | 7,3 | 8,5 |
| 15/5/15 | 20,0 | 13,6 | 10,4 | 12,1 | 13,3 | 12,5 |
| 15/5/30 | 20,0 | 9,4 | 9,5 | 8,4 | 9,0 | 8,5 |
| 15/5/45 | 20,0 | 8,3 | 7,9 | 7,5 | 7,0 | 8,0 |

Le NSGAII-LS effectue systématiquement une PRL à chaque période et n'utilise pas $Ind-MCE$ pour décider de l'application. Le nombre de PRL est donc constant pour le NSGAII-LS (voir tableau 3.9). Par rapport au NSGAII-LS, les autres hybridations ont un nombre d'applications de PRL quasiment divisé par 2. Car l'$Ind - MCE$ n'autorise l'application d'une PRL, que si le nouveau front de Pareto n'a pas beaucoup progressé par rapport à l'ancien. Les petites instances ont plus de PRLs que les grandes, car les algorithmes convergent plus rapidement.

Remarque, une recherche locale n'est pas appliquée sur la totalité des solutions non-dominées de la population pour les NSGAII-(MCE2, MCE5) (voir tableau 3.10). Le nombre de PRL sur les solutions non-dominées augmentent pour les NSGAII hybridés avec la MCE. En déduction, l'utilisation de la MCE augmente la diversification de l'approximation du front de Pareto optimal.

Les hybridations NSGAII-(MCE2, MCE5) effectuent plus de recherches locales sur les solutions dominées que les NSGAII-(MCE1, MCE4), car les solutions dominées sont plus nombreuses dans les $zfe$ (voir le tableau 3.11). Les hybridations NSGAII-(LS, MCE3) n'effectuent pas de recherche locale sur les solutions dominées. Le NSGAII-MCE1 exécute plus de recherches locales sur les solutions

TABLEAU 3.10 – Nombre moyen de recherches locales sur des solutions non-dominées

| Instance | NSGAII - | | | | |
|---|---|---|---|---|---|
| taille | LS | MCE1 | MCE2 | MCE4 | MCE5 |
| 7/3/10 | 11,295 | 14,141 | 11,836 | 13,753 | 8,140 |
| 8/4/8 | 10,110 | 12,284 | 11,088 | 11,667 | 6,541 |
| 10/3/10 | 10,540 | 12,056 | 10,728 | 12,553 | 7,860 |
| 10/4/15 | 11,595 | 14,081 | 12,982 | 16,213 | 10,600 |
| 10/5/10 | 9,785 | 12,866 | 11,907 | 13,628 | 10,625 |
| 10/5/20 | 4,590 | 5,080 | 6,358 | 6,015 | 5,082 |
| 10/5/30 | 3,465 | 4,279 | 4,254 | 5,349 | 4,107 |
| 15/5/15 | 9,780 | 12,333 | 10,912 | 13,414 | 9,917 |
| 15/5/30 | 3,635 | 3,784 | 4,345 | 5,178 | 6,154 |
| 15/5/45 | 3,675 | 3,992 | 3,918 | 5,069 | 4,263 |

TABLEAU 3.11 – Nombre moyen de recherche locale sur les solutions dominées

| Instance | NSGAII - | | | |
|---|---|---|---|---|
| taille | MCE1 | MCE2 | MCE4 | MCE5 |
| 7/3/10 | 84,861 | 141,911 | 53,012 | 133,162 |
| 8/4/8 | 87,525 | 141,309 | 54,469 | 135,622 |
| 10/3/10 | 75,405 | 137,750 | 49,165 | 130,624 |
| 10/4/15 | 72,455 | 136,004 | 51,322 | 127,533 |
| 10/5/10 | 85,102 | 141,186 | 69,295 | 134,800 |
| 10/5/20 | 31,641 | 142,512 | 25,081 | 139,010 |
| 10/5/30 | 25,004 | 144,033 | 20,475 | 136,202 |
| 15/5/15 | 67,177 | 139,271 | 53,018 | 130,440 |
| 15/5/30 | 20,881 | 139,116 | 22,366 | 135,294 |
| 15/5/45 | 17,662 | 140,116 | 19,808 | 133,978 |

dominées que le NSGAII-MCE4, même remarque pour le NSGAII-MCE2 avec le NSGAII-MCE5. Pour déduction, la combinaison de recherches locales dans les $znd$ ou $zfe$ et les $zi$ étale la recherche de l'algorithme dans l'espace de solutions.

Les NSGAII-(MCE3, MCE4, MCE5) sont les seuls algorithmes à générer des solutions dans les zones inexplorées (voir tableau 3.12). Les nombres de solutions créées sont équivalents pour les trois algorithmes. Cette hybridation améliore la diversification des gènes dans la population d'individus.

Les algorithmes NSGAII-(MCE3, MCE4, MCE5) sont plus lents à cause de la génération de solutions dans les $zi$ avant l'application des recherches locales (voir tableau 3.13). Les algorithmes avec des recherches locales sur les solutions des $znd$, NSGAII-(MCE1, MCE4) sont les plus rapides car ils en effectuent sur moins d'individus. La classification par ordre décroissant de rapidité est : NSGAII,

TABLEAU 3.12 – Moyenne des solutions crées dans les zones inexplorées

| Instance | NSGAII - | | |
|---|---|---|---|
| taille | MCE3 | MCE4 | MCE5 |
| 7/3/10 | 34,724 | 33,200 | 34,915 |
| 8/4/8 | 29,422 | 29,848 | 29,848 |
| 10/3/10 | 40,026 | 39,194 | 41,613 |
| 10/4/15 | 63,905 | 63,899 | 67,538 |
| 10/5/10 | 33,905 | 32,335 | 31,659 |
| 10/5/20 | 60,680 | 73,2916 | 54,557 |
| 10/5/30 | 54,018 | 67,362 | 55,197 |
| 15/5/15 | 56,253 | 52,424 | 45,481 |
| 15/5/30 | 77,695 | 74,472 | 59,311 |
| 15/5/45 | 71,384 | 69,120 | 58,520 |

TABLEAU 3.13 – Temps d'exécution (s)

| Instance | NSGAII - | | | | | | |
|---|---|---|---|---|---|---|---|
| taille | | LS | MCE1 | MCE2 | MCE3 | MCE4 | MCE5 |
| 7/3/10 | 3,8 | 6,3 | 8,4 | 9,6 | 10,1 | 10,9 | 12,2 |
| 8/4/8 | 3,7 | 5,5 | 7,6 | 8,9 | 8,4 | 9,1 | 10,6 |
| 10/3/10 | 3,8 | 6,5 | 8,9 | 10,7 | 11,8 | 12,6 | 14,4 |
| 10/4/15 | 5,5 | 12,1 | 14,4 | 16,9 | 17,8 | 19,1 | 20,4 |
| 10/5/10 | 4,7 | 6,4 | 7,7 | 9,0 | 9,2 | 9,6 | 12,5 |
| 10/5/20 | 6,3 | 10,2 | 11,0 | 13,4 | 14,5 | 16,0 | 21,6 |
| 10/5/30 | 8,8 | 17,9 | 18,2 | 23,0 | 22,8 | 19,2 | 28,6 |
| 15/5/15 | 5 | 8,1 | 11,9 | 12,8 | 17,1 | 18,8 | 19,2 |
| 15/5/30 | 7,9 | 12,7 | 18,8 | 28,3 | 27,9 | 28,7 | 27,3 |
| 15/5/45 | 10,7 | 18,2 | 29,7 | 36,9 | 45,1 | 41,9 | 49,4 |

NSGAII-LS, NSGAII-MCE1, NSGAII-MCE2, NSGAII-MCE3, NSGAII-MCE4 et NSGAII-MCE5. Le temps est le seul paramètre où les hybridations avec la MCE sont en désavantage, car des recherches locales sont faites sur plus d'individus.

### 3.4.6 Résultats expérimentaux

Les algorithmes sont répartis en trois groupes pour être comparés, le NSGAII avec le NSGAII-LS, le second est constitué par des hybridations NSGAII-(MCE1, MCE2, MCE3) et le dernier des NSGAII-(MCE4, MCE5). Les meilleurs de chaque groupe sont ensuite comparés entre eux. Les résultats de ces comparaisons sont dans les tableaux 3.14, 3.15 et 3.16, chaque ligne contient

les résultats pour une instance. Chaque ligne est divisée en deux parties, la première contient la moyenne des résultats de $C(Alg_1, Alg_2)$ et la seconde de $C(Alg_2, Alg_1)$. Les colonnes, divisées en trois, contiennent les notes minimum, moyen, et maximum des résultats. Le tableau 3.17 présente les résultats de la métrique H, chaque ligne contient les notes minimum, moyen et maximum du rapport $\frac{H_2}{H_1}$ des surfaces occupées par le front de Pareto des algorithmes $Alg_1$ et $Alg_2$.

La métrique C montre que le NSGAII-MCE3 est dominé par NSGAII-(MCE1, MCE2) (voir tableau 3.14). Les NSGAII-MCE1 et NSGAII-MCE2 obtiennent des résultats équivalents dans les instances $I_1$, $I_2$, $I_3$, $I_4$ et $I_8$. Mais le NSGAII-MCE2 domine le NSGAII-MCE1 sur les grandes instances $I_5$, $I_6$, $I_7$, $I_9$ et $I_{10}$. Le NSGAII-MCE3 obtient ses meilleurs résultats sur les instances de grandes tailles, car il est plus facile de trouver de nouvelles solutions intéressantes dans les grandes instances.

Le NSGAII-LS domine le NSGAII sur toutes les instances (voir le tableau 3.15). Les recherches locales sur les solutions non-dominées accélèrent la convergence du NSGAII. La comparaison entre

TABLEAU 3.14 – Résultats de la métrique C pour les NSGAII-(MCE1, MCE2, MCE3)

| Instance | NSGAII-MCE1/MCE2 | | | NSGAII-MCE1/MCE3 | | | NSGAII-MCE2/MCE3 | | |
|---|---|---|---|---|---|---|---|---|---|
| taille | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 0,000 | 0,067 | 0,167 | 0,000 | 0,155 | 0,385 | 0,000 | 0,206 | 0,461 |
| 7/3/10 | 0,000 | 0,097 | 0,250 | 0,000 | 0,075 | 0,333 | 0,000 | 0,042 | 0,235 |
| $I_2$ | 0,000 | 0,021 | 0,071 | 0,000 | 0,114 | 0,231 | 0,062 | 0,171 | 0,308 |
| 8/4/8 | 0,000 | 0,037 | 0,214 | 0,000 | 0,022 | 0,077 | 0,000 | 0,038 | 0,083 |
| $I_3$ | 0,000 | 0,029 | 0,143 | 0,000 | 0,047 | 0,231 | 0,000 | 0,057 | 0,182 |
| 10/3/10 | 0,000 | 0,026 | 0,167 | 0,000 | 0,014 | 0,071 | 0,000 | 0,022 | 0,154 |
| $I_4$ | 0,000 | 0,087 | 0,200 | 0,000 | 0,103 | 0,333 | 0,000 | 0,140 | 0,308 |
| 10/4/15 | 0,056 | 0,117 | 0,214 | 0,000 | 0,135 | 0,222 | 0,000 | 0,120 | 0,300 |
| $I_5$ | 0,000 | 0,032 | 0,200 | 0,357 | 0,670 | 0,917 | 0,769 | 0,883 | 1,000 |
| 10/5/10 | 0,000 | 0,346 | 0,667 | 0,000 | 0,007 | 0,071 | 0,000 | 0,000 | 0,000 |
| $I_6$ | 0,000 | 0,060 | 0,143 | 0,000 | 0,333 | 0,714 | 0,250 | 0,605 | 1,000 |
| 10/5/20 | 0,250 | 0,534 | 0,800 | 0,000 | 0,216 | 1,000 | 0,000 | 0,056 | 0,222 |
| $I_7$ | 0,000 | 0,062 | 0,400 | 0,000 | 0,302 | 0,750 | 0,333 | 0,679 | 1,000 |
| 10/5/30 | 0,000 | 0,490 | 1,000 | 0,000 | 0,210 | 0,500 | 0,000 | 0,020 | 0,200 |
| $I_8$ | 0,000 | 0,127 | 0,286 | 0,067 | 0,234 | 0,400 | 0,000 | 0,177 | 0,471 |
| 15/5/15 | 0,083 | 0,254 | 0,444 | 0,083 | 0,252 | 0,461 | 0,071 | 0,169 | 0,500 |
| $I_9$ | 0,000 | 0,054 | 0,286 | 0,000 | 0,238 | 0,444 | 0,000 | 0,334 | 0,667 |
| 15/5/30 | 0,000 | 0,252 | 0,667 | 0,000 | 0,212 | 1,000 | 0,000 | 0,153 | 0,500 |
| $I_{10}$ | 0,000 | 0,100 | 0,500 | 0,000 | 0,209 | 0,600 | 0,200 | 0,497 | 1,000 |
| 15/5/45 | 0,000 | 0,487 | 1,000 | 0,000 | 0,220 | 0,750 | 0,000 | 0,145 | 0,500 |

les deux hybridations, NSGAII-(MCE4, MCE5), montre que les deux algorithmes obtiennent des résultats équivalents sur les instances $I_3$, $I_5$ et $I_8$. Pour les autres instances, le NSGAII-MCE5 obtient de meilleurs résultats que le NSGAII-MCE4. Le guidage sur les grandes instances est plus efficace sur les grandes instances, où le NSGAII-MCE5 obtient ses meilleurs résultats.

Le NSGAII-LS est dominé par les hybridations NSGAII-(MCE2, MCE5) selon la métrique C et H (voir tableaux 3.16 et 3.17). Les NSGAII-(MCE2, MCE5) sont plus efficaces pour les grandes instances et le NSGAII-MCE5 obtient de meilleures solutions que le NSGAII-MCE2. Les meilleurs résultats de l'hybridation NSGAII-MCE2 sont sur les instances de $I_5$, $I_6$ et $I_7$. Pour le NSGAII-MCE5, les meilleurs résultats sont sur les instances $I_8$, $I_9$ et $I_{10}$. Faire des recherches locales sur des solutions dans les $zi$ permet au NSGAII-MCE5 d'obtenir de meilleurs résultats que le NSGAII-MCE2 pour de grandes instances. Le meilleur algorithme est le NSGAII-MCE5, mais il est le plus lent, car il effectue le plus de recherches locales. Cependant, le NSGAII-MCE2 domine le NSGAII-LS et consomme moins de temps que le NSGAII-MCE5. Remarque : en ajustant l'exploration de

TABLEAU 3.15 – Les résultats de la métrique C pour la comparaison des algrithmes NSGAII/NSGAII-LS et NSGAII-(MCE4/MCE5)

| Instance | NSGAII/NSGAII-LS | | | NSGAII-MCE4/MCE5 | | |
|---|---|---|---|---|---|---|
| taille | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 0,000 | 0,010 | 0,100 | 0,000 | 0,000 | 0,000 |
| 7/3/10 | 0,176 | 0,412 | 0,786 | 0,000 | 0,126 | 0,312 |
| $I_2$ | 0,000 | 0,008 | 0,077 | 0,000 | 0,000 | 0,000 |
| 8/4/8 | 0,062 | 0,260 | 0,500 | 0,071 | 0,219 | 0,461 |
| $I_3$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 10/3/10 | 0,067 | 0,200 | 0,312 | 0,000 | 0,036 | 0,083 |
| $I_4$ | 0,000 | 0,010 | 0,100 | 0,000 | 0,003 | 0,031 |
| 10/4/15 | 0,167 | 0,348 | 0,550 | 0,000 | 0,235 | 0,375 |
| $I_5$ | 0,000 | 0,030 | 0,111 | 0,000 | 0,070 | 0,231 |
| 10/5/10 | 0,222 | 0,389 | 0,706 | 0,000 | 0,170 | 0,385 |
| $I_6$ | 0,000 | 0,020 | 0,200 | 0,000 | 0,140 | 0,375 |
| 10/5/20 | 0,091 | 0,514 | 1,000 | 0,000 | 0,412 | 1,000 |
| $I_7$ | 0,000 | 0,150 | 0,500 | 0,000 | 0,180 | 0,400 |
| 10/5/30 | 0,000 | 0,596 | 1,000 | 0,000 | 0,307 | 0,600 |
| $I_8$ | 0,000 | 0,025 | 0,125 | 0,000 | 0,134 | 0,308 |
| 15/5/15 | 0,133 | 0,381 | 0,643 | 0,000 | 0,237 | 0,727 |
| $I_9$ | 0,000 | 0,092 | 0,667 | 0,000 | 0,160 | 0,500 |
| 15/5/30 | 0,000 | 0,463 | 1,000 | 0,000 | 0,350 | 0,750 |
| $I_{10}$ | 0,000 | 0,142 | 1,000 | 0,000 | 0,145 | 1,000 |
| 15/5/45 | 0,000 | 0,467 | 1,000 | 0,000 | 0,440 | 1,000 |

TABLEAU 3.16 – Résultats de la comparaison des meilleurs résultats de chaque groupe par la métrique C

| Instance | NSGAII-MCE2/MCE5 | | | NSGAII-LS/MCE2 | | | NSGAII-LS/MCE5 | | |
|---|---|---|---|---|---|---|---|---|---|
| taille | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 0,000 | 0,009 | 0,048 | 0,000 | 0,007 | 0,071 | 0,000 | 0,000 | 0,000 |
| 7/3/10 | 0,000 | 0,173 | 0,417 | 0,125 | 0,231 | 0,364 | 0,083 | 0,320 | 0,583 |
| $I_2$ | 0,000 | 0,005 | 0,050 | 0,000 | 0,021 | 0,071 | 0,000 | 0,011 | 0,100 |
| 8/4/8 | 0,000 | 0,182 | 0,333 | 0,091 | 0,310 | 0,615 | 0,091 | 0,420 | 0,833 |
| $I_3$ | 0,000 | 0,005 | 0,048 | 0,000 | 0,008 | 0,077 | 0,000 | 0,000 | 0,000 |
| 10/3/10 | 0,000 | 0,072 | 0,333 | 0,077 | 0,155 | 0,273 | 0,100 | 0,255 | 0,417 |
| $I_4$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,027 | 0,105 | 0,000 | 0,000 | 0,000 |
| 10/4/15 | 0,176 | 0,258 | 0,450 | 0,222 | 0,365 | 0,667 | 0,000 | 0,041 | 0,231 |
| $I_5$ | 0,053 | 0,163 | 0,267 | 0,000 | 0,006 | 0,059 | 0,000 | 0,000 | 0,000 |
| 10/5/10 | 0,000 | 0,052 | 0,118 | 0,222 | 0,603 | 0,857 | 0,417 | 0,580 | 0,786 |
| $I_6$ | 0,000 | 0,215 | 0,500 | 0,000 | 0,071 | 0,333 | 0,000 | 0,037 | 0,250 |
| 10/5/20 | 0,000 | 0,252 | 0,714 | 0,200 | 0,553 | 1,000 | 0,400 | 0,728 | 1,000 |
| $I_7$ | 0,000 | 0,173 | 0,667 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 10/5/30 | 0,000 | 0,449 | 1,000 | 0,000 | 0,618 | 1,000 | 0,333 | 0,738 | 1,000 |
| $I_8$ | 0,000 | 0,075 | 0,250 | 0,000 | 0,094 | 0,187 | 0,000 | 0,032 | 0,118 |
| 15/5/15 | 0,000 | 0,298 | 0,643 | 0,000 | 0,355 | 0,625 | 0,333 | 0,546 | 0,750 |
| $I_9$ | 0,000 | 0,098 | 0,333 | 0,000 | 0,034 | 0,200 | 0,000 | 0,000 | 0,000 |
| 15/5/30 | 0,250 | 0,555 | 1,000 | 0,000 | 0,418 | 1,000 | 0,333 | 0,747 | 1,000 |
| $I_{10}$ | 0,000 | 0,179 | 1,000 | 0,000 | 0,067 | 0,500 | 0,000 | 0,020 | 0,200 |
| 15/5/45 | 0,000 | 0,295 | 0,500 | 0,000 | 0,558 | 1,000 | 0,333 | 0,663 | 1,000 |

TABLEAU 3.17 – Résultats de la métrique H pour les meilleurs algorithmes de chaque groupe

| Instance | NSGAII-MCE2/MCE5 | | | NSGAII-LS/MCE2 | | | NSGAII-LS/MCE5 | | |
|---|---|---|---|---|---|---|---|---|---|
| Index | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 0,907 | 0,934 | 0,963 | 0,947 | 0,978 | 1,014 | 0,886 | 0,915 | 0,953 |
| $I_2$ | 0,868 | 0,892 | 0,948 | 0,968 | 0,991 | 1,011 | 0,867 | 0,897 | 0,963 |
| $I_3$ | 0,876 | 0,907 | 0,930 | 0,952 | 0,989 | 1,015 | 0,852 | 0,895 | 0,933 |
| $I_4$ | 0,854 | 0,881 | 0,911 | 0,959 | 0,986 | 1,019 | 0,843 | 0,866 | 0,898 |
| $I_5$ | 0,992 | 1,001 | 1,019 | 0,961 | 0,979 | 1,011 | 0,961 | 0,980 | 1,003 |
| $I_6$ | 0,980 | 0,994 | 1,025 | 0,958 | 0,979 | 1,011 | 0,909 | 0,970 | 1,018 |
| $I_7$ | 0,961 | 0,988 | 1,010 | 0,958 | 0,977 | 0,994 | 0,962 | 0,980 | 0,993 |
| $I_8$ | 0,973 | 0,993 | 1,027 | 0,957 | 0,990 | 1,026 | 0,952 | 0,985 | 1,010 |
| $I_9$ | 0,965 | 0,983 | 0,999 | 0,968 | 0,987 | 1,009 | 0,951 | 0,969 | 0,981 |
| $I_{10}$ | 0,975 | 0,998 | 1,012 | 0,975 | 0,988 | 1,005 | 0,970 | 0,988 | 1,007 |

l'espace de recherche du NSGAII, par intensification et diversification sur l'espace uni-dimensionnel, les résultats sont améliorés.

### 3.4.7 Résultats des comparaisons des algorithmes hybridés

La MCE est maintenant adaptée à d'autres algorithmes pour résoudre le FJSP avec les mêmes objectifs. Dans la section 3.4.5, l'hybridation qui donne les meilleurs résultats, applique une recherche locale sur les solutions dans les $zfe$ et celles créées dans les $zi$ (NSGAII-MCE5). Cette hybridation est adaptée à cinq autres algorithmes (SPEA2 (Zitzler et al., 2001), MOAIS (Yoo and Hajela, 1999), MOACO (Mariano et al., 1999), MOPSO (Coello Coello and Lechuga, 2002) et MOABC (Pham and Ghanbarzadeh, 2007)). Les algorithmes comparés sont fréquemment rencontrés dans la littérature et adaptés pour le FJSP (voir section 3.3.1).

La métrique C et le nombre de solutions dans le front de Pareto sont utilisés comme métriques lors de la comparaison. Les algorithmes sont répartis en deux groupes, le premier groupe contient les algorithmes évolutionnaires et apparentés (NSGAII, SPEA2 et MOAIS) et le deuxième groupe inclut les trois autres algorithmes (MOACO, MOPSO et MOABC). Dans chaque groupe, les algorithmes sont comparés entre eux par la métrique C. Les meilleurs des deux groupes sont ensuite comparés entre eux.

Les résultats sont présentés en trois temps, en premier, les algorithmes sont comparés entre eux sans hybridation ; dans un deuxième temps les algorithmes sont comparés avec leur hybridation et en dernier, les algorithmes hybridés sont comparés entre eux.

Les tableaux 3.18, 3.22, 3.26, 3.28, 3.30, 3.34 et 3.38, contiennent les résultats de la métrique C. Chaque colonne contient les résultats de la comparaison entre deux algorithmes, chacune est divisée en trois parties contenant les notes minimum, moyenne et maximum obtenues. Les résultats d'une instance sont contenus dans une ligne qui est divisée en deux parties, la première contient les résultats de $C(Alg_1, Alg_2)$ et la deuxième de $C(Alg_2, Alg_1)$. Les tableaux 3.19, 3.23, 3.27, 3.29, 3.31, 3.35 et 3.39, contiennent les résultats de la métrique H. Chaque colonne contient, sur une ligne, les notes minimum, moyenne et maximum du rapport $\frac{H_2}{H_1}$ obtenues par les algorithmes $Alg_1$ et $Alg_2$ sur une instance. Les tableaux 3.20, 3.24, 3.32, 3.36, 3.21, 3.25, 3.33 et 3.37 contiennent les résultats de la métrique de diversification et du nombre de solutions non-dominées, les colonnes contiennent les résultats minimum, moyen et maximum pour les algorithmes et les lignes, ceux pour les instances.

Dans le premier groupe (voir tableaux 3.18 et 3.19), le NSGAII et le SPEA2 obtiennent des résultats équivalents. Par contre, le MOAIS domine largement le NSGAII et le SPEA2. Les plus forts écarts entre le MOAIS et les autres algorithmes sont sur les instances de grande taille $I_6$, $I_7$, $I_9$ et $I_{10}$.

TABLEAU 3.18 – Résultats de la métrique C pour la comparaison des algorithmes évolutionnaires

| | NSGAII/SPEA2 | | | NSGAII/MOAIS | | | SPEA2/MOAIS | | |
|---|---|---|---|---|---|---|---|---|---|
| Instance | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 0,231 | 0,546 | 0,733 | 0,000 | 0,010 | 0,050 | 0,000 | 0,016 | 0,067 |
| 7/3/10 | 0,214 | 0,511 | 0,722 | 0,800 | 0,951 | 1,000 | 0,812 | 0,932 | 1,000 |
| $I_2$ | 0,143 | 0,300 | 0,500 | 0,647 | 0,798 | 0,895 | 0,000 | 0,038 | 0,231 |
| 8/4/8 | 0,250 | 0,543 | 0,765 | 0,600 | 0,813 | 1,000 | 0,625 | 0,814 | 0,937 |
| $I_3$ | 0,133 | 0,322 | 0,600 | 0,000 | 0,056 | 0,182 | 0,000 | 0,102 | 0,283 |
| 10/3/10 | 0,231 | 0,546 | 0,733 | 0,538 | 0,769 | 0,937 | 0,471 | 0,663 | 0,857 |
| $I_4$ | 0,111 | 0,388 | 0,667 | 0,000 | 0,011 | 0,056 | 0,000 | 0,005 | 0,053 |
| 10/4/15 | 0,167 | 0,515 | 0,765 | 0,823 | 0,958 | 1,000 | 0,765 | 0,949 | 1,000 |
| $I_5$ | 0,118 | 0,436 | 0,591 | 0,000 | 0,115 | 0,300 | 0,000 | 0,140 | 0,368 |
| 10/5/10 | 0,235 | 0,436 | 0,778 | 0,588 | 0,751 | 0,944 | 0,471 | 0,763 | 0,944 |
| $I_6$ | 0,000 | 0,319 | 0,667 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 10/5/20 | 0,000 | 0,479 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 |
| $I_7$ | 0,000 | 0,495 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 10/5/30 | 0,000 | 0,387 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 |
| $I_8$ | 0,077 | 0,463 | 0,773 | 0,000 | 0,096 | 0,412 | 0,000 | 0,132 | 0,412 |
| 15/5/15 | 0,077 | 0,434 | 0,583 | 0,600 | 0,833 | 1,000 | 0,611 | 0,778 | 1,000 |
| $I_9$ | 0,125 | 0,361 | 0,625 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 15/5/30 | 0,000 | 0,412 | 0,833 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 |
| $I_{10}$ | 0,000 | 0,357 | 0,600 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 15/5/45 | 0,000 | 0,487 | 0,857 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 |

TABLEAU 3.19 – Résultats de la métrique H pour la comparaison des algorithmes évolutionnaires

| Instance | NSGAII/SPEA2 | | | NSGAII/MOAIS | | | SPEA2/MOAIS | | |
|---|---|---|---|---|---|---|---|---|---|
| Index | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 0,974 | 0.994 | 1,026 | 0,889 | 0,929 | 0,962 | 0,902 | 0,937 | 0,981 |
| $I_2$ | 0,984 | 0,995 | 1,021 | 0,943 | 0,981 | 1,026 | 0,961 | 0,985 | 1,033 |
| $I_3$ | 0,971 | 0,988 | 1,002 | 0,944 | 0,986 | 1,033 | 0,959 | 1,007 | 1,037 |
| $I_4$ | 0,973 | 0,996 | 1,019 | 0,904 | 0,928 | 0,972 | 0,880 | 0,935 | 1,120 |
| $I_5$ | 0,970 | 0,999 | 1,014 | 0,946 | 0,981 | 1,002 | 0,955 | 0,981 | 1,005 |
| $I_6$ | 0,949 | 0,997 | 1,031 | 0,855 | 0,898 | 0,945 | 0,848 | 0,894 | 0,938 |
| $I_7$ | 0,982 | 1,002 | 1,011 | 0,860 | 0,909 | 0,936 | 0,888 | 0,911 | 0,947 |
| $I_8$ | 0,981 | 1,000 | 1,023 | 0,937 | 0,971 | 1,001 | 0,938 | 0,972 | 1,003 |
| $I_9$ | 0,979 | 0,997 | 1,011 | 0,845 | 0,969 | 0,892 | 0,856 | 0,878 | 0,901 |
| $I_{10}$ | 0,985 | 0,998 | 1,019 | 0,854 | 0,897 | 0,977 | 0,948 | 0,905 | 0,948 |

TABLEAU 3.20 – Résultat pour la métrique DM pour les algorithmes évolutionnaires

| Instance | NSGA2 | | | SPEA2 | | | MOAIS | | |
|---|---|---|---|---|---|---|---|---|---|
| Index | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 78,102 | 102,706 | 114,970 | 87,863 | 103,253 | 118,870 | 79,044 | 103,484 | 125,391 |
| $I_2$ | 66,219 | 92,534 | 114,42 | 73,348 | 85,382 | 102,772 | 38,588 | 70,774 | 101,43 |
| $I_3$ | 91,918 | 107,446 | 133,955 | 105,679 | 123,338 | 142,899 | 49,366 | 91,988 | 129,988 |
| $I_4$ | 105,759 | 141,987 | 173,367 | 115,204 | 146,712 | 170,625 | 101,651 | 134,991 | 191,857 |
| $I_5$ | 84,427 | 110,318 | 130,415 | 102,103 | 111,052 | 122,372 | 78,428 | 94,387 | 113,044 |
| $I_6$ | 46,680 | 126,273 | 199,361 | 60,083 | 113,837 | 176,553 | 55,534 | 110,686 | 149,623 |
| $I_7$ | 91,433 | 207,580 | 435,033 | 11,358 | 159,651 | 326,256 | 81,780 | 171,960 | 391,061 |
| $I_8$ | 110,635 | 147,837 | 202,783 | 103,073 | 150,457 | 200,242 | 85,697 | 113,188 | 146,455 |
| $I_9$ | 131,522 | 256,049 | 499,289 | 68,935 | 249,149 | 458,070 | 72,993 | 162,717 | 206,175 |
| $I_{10}$ | 333,542 | 509,816 | 646,250 | 365,066 | 522,558 | 680,354 | 84,048 | 266,046 | 506,009 |

TABLEAU 3.21 – Nombre de solutions non-dominées pour les algorithmes évolutionnaires

| Instance | NSGAII | | | SPEA2 | | | MOAIS | | |
|---|---|---|---|---|---|---|---|---|---|
| Index | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 14 | 16,7 | 20 | 14 | 17,2 | 24 | 17 | 19,3 | 23 |
| $I_2$ | 13 | 15,3 | 17 | 13 | 15,2 | 19 | 15 | 15,8 | 18 |
| $I_3$ | 13 | 14,1 | 16 | 13 | 15,1 | 17 | 11 | 13,4 | 17 |
| $I_4$ | 12 | 16,9 | 20 | 13 | 17,7 | 20 | 17 | 21,6 | 26 |
| $I_5$ | 16 | 18,2 | 21 | 15 | 18,5 | 21 | 17 | 20,6 | 25 |
| $I_6$ | 2 | 6,4 | 10 | 4 | 6,1 | 8 | 8 | 11,2 | 14 |
| $I_7$ | 3 | 4,9 | 6 | 1 | 4,6 | 9 | 3 | 6,2 | 10 |
| $I_8$ | 12 | 16,2 | 20 | 13 | 16,5 | 20 | 15 | 17,8 | 22 |
| $I_9$ | 4 | 7,3 | 11 | 3 | 7,3 | 12 | 7 | 9,4 | 11 |
| $I_{10}$ | 5 | 7,0 | 10 | 4 | 6,8 | 11 | 3 | 5,0 | 8 |

Logiquement, les algorithmes évolutionnaires obtiennent des fronts de Pareto plus étalés sur les grandes instances (voir tableau 3.20). Sur la métrique DM, les trois algorithmes obtiennent des notes équivalentes mais le MOAIS a souvent un front de Pareto plus resserré que les NSGAII et SPEA2. Le MOAIS obtient plus de solutions dans le front de Pareto optimal, mais les nombres de solutions non-dominées pour les trois algorithmes restent comparables (voir tableau 3.21). Les algorithmes obtiennent plus de solutions non-dominées sur les petites instances $I_1$, $I_2$, $I_3$, $I_4$, $I_5$ et $I_8$.

TABLEAU 3.22 – Résultats de la métrique C pour la comparaison des MOACO, MOPSO et MOABC

| | MOACO/MOPSO | | | MOACO/MOABC | | | MOPSO/MOABC | | |
|---|---|---|---|---|---|---|---|---|---|
| Instance | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 0,176 | 0,294 | 0,435 | 0,000 | 0,000 | 0,000 | 0,000 | 0,016 | 0,056 |
| 7/3/10 | 0,125 | 0,347 | 0,500 | 0,937 | 0,994 | 1,000 | 0,571 | 0,643 | 0,762 |
| $I_2$ | 0,286 | 0,377 | 0,500 | 0,000 | 0,000 | 0,000 | 0,000 | 0,033 | 0,125 |
| 8/4/8 | 0,071 | 0,195 | 0,357 | 0,923 | 0,985 | 1,000 | 0,450 | 0,528 | 0,625 |
| $I_3$ | 0,278 | 0,435 | 0,524 | 0,000 | 0,000 | 0,000 | 0,000 | 0,022 | 0,077 |
| 10/3/10 | 0,077 | 0,251 | 0,385 | 0,833 | 0,925 | 1,000 | 0,500 | 0,585 | 0,667 |
| $I_4$ | 0,087 | 0,239 | 0,391 | 0,000 | 0,000 | 0,000 | 0,000 | 0,005 | 0,050 |
| 10/4/15 | 0,210 | 0,421 | 0,929 | 0,950 | 0,995 | 1,000 | 0,461 | 0,557 | 0,615 |
| $I_5$ | 0,000 | 0,235 | 0,423 | 0,000 | 0,004 | 0,045 | 0,000 | 0,027 | 0,087 |
| 10/5/10 | 0,412 | 0,623 | 0,882 | 0,944 | 0,994 | 1,000 | 0,833 | 0,901 | 1,000 |
| $I_6$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 10/5/20 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 |
| $I_7$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 10/5/30 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 |
| $I_8$ | 0,000 | 0,203 | 0,529 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 15/5/15 | 0,214 | 0,672 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 |
| $I_9$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 15/5/30 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 |
| $I_{10}$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 15/5/45 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 |

TABLEAU 3.23 – Résultats de la métrique H pour la comparaison des MOACO, MOPSO et MOABC

| Instance | MOACO/MOPSO | | | MOACO/MOABC | | | MOPSO/MOABC | | |
|---|---|---|---|---|---|---|---|---|---|
| Index | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 0,906 | 0,941 | 0,989 | 0,849 | 0,873 | 0,896 | 0,927 | 0,950 | 0,971 |
| $I_2$ | 0,883 | 0,919 | 0,964 | 0,910 | 0,926 | 0,949 | 0,985 | 1,029 | 1,062 |
| $I_3$ | 0,927 | 0,987 | 1,072 | 0,893 | 0,924 | 0,965 | 0,867 | 0,955 | 1,027 |
| $I_4$ | 0,844 | 0,912 | 0,957 | 0,797 | 0,834 | 0,870 | 0,895 | 0,956 | 1,046 |
| $I_5$ | 0,976 | 0,983 | 0,999 | 0,926 | 0,935 | 0,942 | 0,943 | 0,952 | 0,960 |
| $I_6$ | 0,878 | 0,897 | 0,927 | 0,763 | 0,801 | 0,831 | 0,868 | 0,902 | 0,922 |
| $I_7$ | 0,856 | 0,915 | 0,974 | 0,744 | 0,779 | 0,858 | 0,894 | 0,927 | 0,959 |
| $I_8$ | 0,946 | 0,979 | 1,017 | 0,858 | 0,869 | 0,882 | 0,861 | 0,887 | 0,916 |
| $I_9$ | 0,854 | 0,907 | 0,947 | 0,690 | 0,776 | 0,824 | 0,823 | 0,882 | 0,932 |
| $I_{10}$ | 0,834 | 0,903 | 0,982 | 0,741 | 0,781 | 0,821 | 0,899 | 0,943 | 0,972 |

Selon la métrique C pour toutes les instances, le classement du deuxième groupe du plus au moins performant est : MOABC, MOPSO et MOACO (voir tableau 3.22 et 3.23). Le MOABC obtient ses meilleurs résultats sur les grandes instances $I_5$, $I_6$, $I_7$, $I_8$, $I_9$ et $I_{10}$. Le MOACO et le MOPSO obtiennent des résultats équivalents sur les instances $I_1$, $I_2$, $I_3$ et $I_4$, par contre sur les autres instances, le MOPSO domine nettement le MOACO.

TABLEAU 3.24 – Résultat pour la métrique DM pour les MOACO, MOPSO et MOABC

| Instance | MOACO | | | MOPSO | | | MOABC | | |
|---|---|---|---|---|---|---|---|---|---|
| Index | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 80,951 | 96,620 | 125,164 | 120,412 | 141,385 | 159,875 | 83,229 | 99,009 | 118,528 |
| $I_2$ | 51,913 | 74,667 | 101,430 | 121,012 | 133,655 | 149,278 | 61,604 | 78,737 | 89,258 |
| $I_3$ | 102,820 | 126,332 | 155,168 | 135,952 | 145,395 | 160,378 | 98,610 | 110,892 | 127,656 |
| $I_4$ | 94,340 | 150,986 | 221,817 | 201,398 | 239,764 | 265,021 | 110,635 | 132,656 | 154,353 |
| $I_5$ | 92,434 | 108,270 | 138,654 | 94,636 | 108,597 | 126,602 | 84,764 | 107,412 | 145,338 |
| $I_6$ | 40,988 | 98,011 | 155,461 | 0,000 | 59,456 | 91,236 | 52,887 | 77,164 | 125,475 |
| $I_7$ | 0,000 | 140,196 | 358,869 | 0,000 | 24,419 | 78,626 | 58,686 | 95,246 | 125,252 |
| $I_8$ | 118,920 | 142,365 | 165,529 | 120,599 | 135,850 | 154,800 | 115,430 | 133,725 | 157,648 |
| $I_9$ | 71,162 | 126,097 | 196,499 | 0,000 | 34,234 | 74,465 | 64,630 | 142,163 | 511,097 |
| $I_{10}$ | 87,561 | 257,906 | 494,170 | 0,000 | 23,659 | 118,013 | 13,416 | 122,254 | 245,165 |

TABLEAU 3.25 – Nombre de solutions non-dominées pour les MOACO, MOPSO et MOABC

| Instance | MOACO | | | MOPSO | | | MOABC | | |
|---|---|---|---|---|---|---|---|---|---|
| Index | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 11 | 14,4 | 16 | 15 | 17,6 | 20 | 17 | 19,1 | 21 |
| $I_2$ | 9 | 12,5 | 15 | 19 | 20,5 | 21 | 14 | 15,0 | 16 |
| $I_3$ | 12 | 13,7 | 16 | 14 | 17,6 | 19 | 11 | 12,3 | 15 |
| $I_4$ | 12 | 15,5 | 20 | 14 | 17,1 | 21 | 15 | 17,8 | 22 |
| $I_5$ | 16 | 18,4 | 23 | 17 | 20,1 | 24 | 20 | 22,0 | 24 |
| $I_6$ | 3 | 6,3 | 10 | 6 | 9,5 | 13 | 9 | 13,4 | 18 |
| $I_7$ | 1 | 5,0 | 8 | 1 | 3,0 | 5 | 2 | 3,4 | 5 |
| $I_8$ | 13 | 16,7 | 19 | 8 | 13,1 | 17 | 15 | 19,5 | 23 |
| $I_9$ | 2 | 4,4 | 7 | 1 | 2,4 | 4 | 5 | 9,6 | 15 |
| $I_{10}$ | 2 | 5,8 | 10 | 1 | 1,6 | 4 | 1 | 2,1 | 4 |

Sur la métrique DM, les algorithmes du deuxième groupe, MOACO, MOPSO et MOABC, obtiennent des notes équivalentes sur les petites instances (voir tableau 3.24). Par contre, sur les grandes instances, le MOACO obtient les meilleures notes, suivi par le MOABC. Le MOPSO a les plus mauvaises notes à la métrique de diversification sur les instances : $I_6$, $I_7$, $I_9$ et $I_{10}$. Le bilan pour le nombre d'individus non-dominés est moins net pour le deuxième groupe (voir tableau 3.25).

Les algorithmes comme pour le premier groupe ont plus de solutions dans le front de Pareto optimal sur les petites instances $I_1$, $I_2$, $I_3$, $I_4$, $I_5$ et $I_8$. Le MOACO obtient plus de solutions non-dominées sur les grandes instances $I_7$ et $I_{10}$ que les autres algorithmes. Sur les instances $I_2$ et $I_3$, le MOPSO a le plus de solutions non-dominées. Sur les autres instances, le MOABC obtient plus d'individus non-dominés.

L'hybridation Alg-MCE5 améliore les performances de tous les algorithmes (voir tableaux 3.26, 3.27, 3.28 et 3.29). L'amélioration est plus marquée sur les grandes instances $I_6$, $I_7$, $I_9$ et $I_{10}$. Les NSGAII, SPEA2 et MOACO sont les algorithmes les plus sensibles à l'hybridation. Par contre, l'hybridation est moins efficace sur le MOABC. Selon la DM, l'étalement du front de Pareto reste équivalent entre les algorithmes classiques et hybridés sur les petites instances : $I_1$, $I_2$, $I_3$ et $I_4$ (voir tableaux 3.20, 3.24, 3.32 et 3.36). Par contre, sur les grandes instances, les solutions non-dominées sont beaucoup plus resserrées pour les algorithmes hybridés, notamment pour le SPEA2 sur les instances : $I_6$, $I_7$, $I_9$ et $I_{10}$. La diversité sur le front de Pareto des algorithmes est modifiée par l'hybridation, mais le nombre de solutions non-dominées reste similaire pour chaque algorithme sur chaque instance (voir tableaux 3.21, 3.25, 3.33 et 3.37).

Pour le groupe des algorithmes évolutionnaires et apparentés, le MOAIS-MCE5 obtient les meilleurs résultats par rapport au NSGAII-MCE5 et au SPEA2-MCE5 selon les métriques C et H (voir tableau 3.30 et 3.31). Par contre, le MOAIS-MCE5 domine moins les deux autres algorithmes sur les instances de petites tailles $I_1$, $I_2$, $I_3$ et $I_4$ et il est même équivalent sur l'instance $I_8$. L'hybridation profite plus au NSGAII-MCE5 car il a des résultats équivalents au MOAIS-MCE5 sur les petites instances $I_1$, $I_3$ et $I_4$.

TABLEAU 3.26 – Comparaison par la métrique C des algorithmes évolutionnaires et leur hybridation MCE5

| | NSGAII-(./MCE5) | | | SPEA2-(./MCE5) | | | MOAIS-(./MCE5) | | |
|---|---|---|---|---|---|---|---|---|---|
| Instance | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,033 | 0,158 |
| 7/3/10 | 1,000 | 1,000 | 1,000 | 0,929 | 0,981 | 1,000 | 0,667 | 0,872 | 1,000 |
| $I_2$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,007 | 0,067 | 0,000 | 0,006 | 0,062 |
| 8/4/8 | 0,937 | 0,987 | 1,000 | 0,923 | 0,960 | 1,000 | 0,823 | 0,960 | 1,000 |
| $I_3$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,013 | 0,067 | 0,000 | 0,013 | 0,133 |
| 10/3/10 | 1,000 | 1,000 | 1,000 | 0,882 | 0,964 | 1,000 | 0,765 | 0,944 | 1,000 |
| $I_4$ | 0,000 | 0,013 | 0,103 | 0,000 | 0,000 | 0,000 | 0,000 | 0,125 | 0,423 |
| 10/4/15 | 0,823 | 0,947 | 1,000 | 0,778 | 0,954 | 1,000 | 0,556 | 0,806 | 0,944 |
| $I_5$ | 0,000 | 0,005 | 0,050 | 0,000 | 0,004 | 0,043 | 0,000 | 0,000 | 0,000 |
| 10/5/10 | 0,895 | 0,973 | 1,000 | 0,900 | 0,952 | 1,000 | 0,789 | 0,964 | 1,000 |
| $I_6$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,007 | 0,067 |
| 10/5/20 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 0,800 | 0,958 | 1,000 |
| $I_7$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 10/5/30 | 1,000 | 1,000 | 1,000 | 0,833 | 0,944 | 1,000 | 0,800 | 0,980 | 1,000 |
| $I_8$ | 0,000 | 0,038 | 0,111 | 0,000 | 0,048 | 0,118 | 0,000 | 0,154 | 0,333 |
| 15/5/15 | 0,667 | 0,873 | 1,000 | 0,687 | 0,850 | 1,000 | 0,333 | 0,653 | 0,941 |
| $I_9$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,129 | 0,571 |
| 15/5/30 | 1,000 | 1,000 | 1,000 | 0,500 | 0,839 | 1,000 | 0,429 | 0,774 | 1,000 |
| $I_{10}$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,040 | 0,400 |
| 15/5/45 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 0,333 | 0,919 | 1,000 |

TABLEAU 3.27 – Comparaison par la métrique H des algorithmes évolutionnaires et leur hybridation MCE5

| Instance | NSGAII-(./MCE5) | | | SPEA2-(./MCE5) | | | MOAIS-(./MCE5) | | |
|---|---|---|---|---|---|---|---|---|---|
| Index | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 0,841 | 0,863 | 0,877 | 0,903 | 0,920 | 0,936 | 0,928 | 0,965 | 0,995 |
| $I_2$ | 0,828 | 0,870 | 0,898 | 0,933 | 0,954 | 0,972 | 0,920 | 0,963 | 0,984 |
| $I_3$ | 0,842 | 0,871 | 0,900 | 0,936 | 0,949 | 0,965 | 0,940 | 0,962 | 0,981 |
| $I_4$ | 0,816 | 0,838 | 0,858 | 0,912 | 0,933 | 0,972 | 0,949 | 0,973 | 0,988 |
| $I_5$ | 0,933 | 0,942 | 0,959 | 0,930 | 0,949 | 0,966 | 0,931 | 0,950 | 0,964 |
| $I_6$ | 0,912 | 0,935 | 0,985 | 0,881 | 0,927 | 0,974 | 0,940 | 0,964 | 0,982 |
| $I_7$ | 0,916 | 0,950 | 0,988 | 0,949 | 0,971 | 0,988 | 0,964 | 0,977 | 0,984 |
| $I_8$ | 0,927 | 0,960 | 0,983 | 0,929 | 0,960 | 0,981 | 0,950 | 0,984 | 1,001 |
| $I_9$ | 0,913 | 0,927 | 0,940 | 0,943 | 0,974 | 1,020 | 0,944 | 0,979 | 1,001 |
| $I_{10}$ | 0,944 | 0,959 | 0,976 | 0,885 | 0,929 | 0,963 | 0,977 | 0,985 | 0,999 |

TABLEAU 3.28 – Comparaison par la métrique C des méthodes du second groupe et leur hybridation MCE5

| | MOACO-(./MCE5) | | | MOPSO-(./MCE5) | | | MOABC-(./MCE5) | | |
|---|---|---|---|---|---|---|---|---|---|
| Instance | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,106 | 0,235 | 0,000 | 0,167 | 0,428 |
| 7/3/10 | 0,929 | 0,987 | 1,000 | 0,667 | 0,818 | 0,947 | 0,500 | 0,635 | 0,789 |
| $I_2$ | 0,000 | 0,016 | 0,100 | 0,000 | 0,115 | 0,235 | 0,071 | 0,203 | 0,375 |
| 8/4/8 | 0,778 | 0,946 | 1,000 | 0,625 | 0,820 | 1,000 | 0,375 | 0,515 | 0,714 |
| $I_3$ | 0,000 | 0,000 | 0,000 | 0,062 | 0,140 | 0,250 | 0,000 | 0,161 | 0,364 |
| 10/3/10 | 0,917 | 0,969 | 1,000 | 0,714 | 0,822 | 0,905 | 0,333 | 0,605 | 0,917 |
| $I_4$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,168 | 0,444 | 0,062 | 0,133 | 0,250 |
| 10/4/15 | 0,917 | 0,979 | 1,000 | 0,625 | 0,791 | 1,000 | 0,579 | 0,743 | 0,900 |
| $I_5$ | 0,000 | 0,037 | 0,100 | 0,000 | 0,154 | 0,600 | 0,048 | 0,191 | 0,435 |
| 10/5/10 | 0,762 | 0,892 | 0,956 | 0,273 | 0,719 | 0,913 | 0,200 | 0,598 | 0,773 |
| $I_6$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,109 | 0,461 | 0,000 | 0,102 | 0,364 |
| 10/5/20 | 1,000 | 1,000 | 1,000 | 0,000 | 0,701 | 1,000 | 0,625 | 0,827 | 1,000 |
| $I_7$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,100 | 0,750 |
| 10/5/30 | 1,000 | 1,000 | 1,000 | 0,857 | 0,986 | 1,000 | 0,000 | 0,778 | 1,000 |
| $I_8$ | 0,000 | 0,121 | 0,454 | 0,000 | 0,045 | 0,214 | 0,000 | 0,178 | 0,381 |
| 15/5/15 | 0,579 | 0,783 | 0,895 | 0,526 | 0,875 | 1,000 | 0,444 | 0,672 | 0,958 |
| $I_9$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,050 | 0,500 | 0,000 | 0,167 | 0,667 |
| 15/5/30 | 1,000 | 1,000 | 1,000 | 0,200 | 0,810 | 1,000 | 0,286 | 0,770 | 1,000 |
| $I_{10}$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,117 | 0,667 | 0,000 | 0,120 | 0,500 |
| 15/5/45 | 1,000 | 1,000 | 1,000 | 0,000 | 0,650 | 1,000 | 0,167 | 0,690 | 1,000 |

TABLEAU 3.29 – Comparaison par la métrique H des méthodes du second groupe et leur hybridation MCE5

| Instance | MOACO-(./MCE5) | | | MOPSO-(./MCE5) | | | MOABC-(./MCE5) | | |
|---|---|---|---|---|---|---|---|---|---|
| Index | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 0,860 | 0,897 | 0,922 | 0,906 | 0,952 | 0,993 | 0,971 | 0,987 | 1,005 |
| $I_2$ | 0,913 | 0,940 | 0,957 | 0,941 | 0,960 | 0,977 | 0,972 | 0,989 | 1,013 |
| $I_3$ | 0,909 | 0,929 | 0,945 | 0,868 | 0,914 | 0,947 | 0,945 | 0,990 | 1,012 |
| $I_4$ | 0,851 | 0,884 | 0,916 | 0,885 | 0,947 | 1,004 | 0,955 | 0,980 | 0,994 |
| $I_5$ | 0,932 | 0,951 | 0,973 | 0,966 | 0,979 | 1,009 | 0,981 | 0,994 | 1,003 |
| $I_6$ | 0,859 | 0,907 | 0,939 | 0,906 | 0,960 | 1,000 | 0,973 | 0,985 | 0,999 |
| $I_7$ | 0,880 | 0,919 | 0,954 | 0,939 | 0,972 | 0,992 | 0,986 | 0,992 | 1,006 |
| $I_8$ | 0,947 | 0,970 | 0,994 | 0,924 | 0,945 | 0,996 | 0,968 | 0,985 | 1,001 |
| $I_9$ | 0,897 | 0,921 | 0,941 | 0,949 | 0,978 | 1,002 | 0,972 | 0,989 | 1,019 |
| $I_{10}$ | 0,919 | 0,941 | 0,968 | 0,953 | 0,991 | 1,007 | 0,981 | 0,993 | 1,001 |

TABLEAU 3.30 – Résultats de la métrique C des algorithmes évolutionnaires hybridés par MCE5

| | NSGAII/SPEA2 | | | NSGAII/MOAIS | | | SPEA2/MOAIS | | |
|---|---|---|---|---|---|---|---|---|---|
| Instance | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 0,333 | 0,527 | 0,750 | 0,167 | 0,308 | 0,421 | 0,091 | 0,177 | 0,267 |
| 7/3/10 | 0,087 | 0,188 | 0,250 | 0,250 | 0,379 | 0,522 | 0,412 | 0,601 | 0,750 |
| $I_2$ | 0,333 | 0,502 | 0,750 | 0,000 | 0,198 | 0,375 | 0,000 | 0,124 | 0,267 |
| 8/4/8 | 0,050 | 0,228 | 0,350 | 0,261 | 0,451 | 0,700 | 0,400 | 0,702 | 0,937 |
| $I_3$ | 0,533 | 0,593 | 0,687 | 0,182 | 0,312 | 0,471 | 0,000 | 0,218 | 0,500 |
| 10/3/10 | 0,053 | 0,194 | 0,286 | 0,158 | 0,322 | 0,476 | 0,286 | 0,564 | 0,750 |
| $I_4$ | 0,222 | 0,533 | 0,789 | 0,115 | 0,352 | 0,522 | 0,000 | 0,212 | 0,611 |
| 10/4/15 | 0,069 | 0,173 | 0,345 | 0,143 | 0,272 | 0,536 | 0,235 | 0,711 | 0,905 |
| $I_5$ | 0,316 | 0,497 | 0,667 | 0,045 | 0,314 | 0,773 | 0,045 | 0,242 | 0,454 |
| 10/5/10 | 0,222 | 0,345 | 0,444 | 0,150 | 0,540 | 0,823 | 0,278 | 0,595 | 0,900 |
| $I_6$ | 0,000 | 0,410 | 0,778 | 0,015 | 0,062 | 1,087 | 0,000 | 0,041 | 0,154 |
| 10/5/20 | 0,000 | 0,356 | 1,000 | 0,920 | 0,942 | 0,985 | 0,600 | 0,881 | 1,000 |
| $I_7$ | 0,000 | 0,642 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 10/5/30 | 0,000 | 0,254 | 0,750 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 |
| $I_8$ | 0,250 | 0,451 | 0,632 | 0,105 | 0,424 | 0,647 | 0,210 | 0,463 | 0,789 |
| 15/5/15 | 0,214 | 0,435 | 0,588 | 0,077 | 0,442 | 0,882 | 0,187 | 0,373 | 0,650 |
| $I_9$ | 0,333 | 0,785 | 1,000 | 0,000 | 0,009 | 0,091 | 0,000 | 0,000 | 0,000 |
| 15/5/30 | 0,000 | 0,093 | 0,400 | 0,750 | 0,975 | 1,000 | 1,000 | 1,000 | 1,000 |
| $I_{10}$ | 0,571 | 0,812 | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 15/5/45 | 0,000 | 0,000 | 0,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 |

TABLEAU 3.31 – Résultats de la métrique H des algorithmes évolutionnaires hybridés par MCE5

| Instance | NSGAII/SPEA2 | | | NSGAII/MOAIS | | | SPEA2/MOAIS | | |
|---|---|---|---|---|---|---|---|---|---|
| Index | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 1,046 | 1,068 | 1,100 | 0,909 | 0,943 | 0,975 | 0,978 | 0,986 | 1,003 |
| $I_2$ | 1,047 | 1,100 | 1,147 | 0,878 | 0,904 | 0,974 | 0,974 | 0,993 | 1,015 |
| $I_3$ | 1,038 | 1,067 | 1,087 | 0,841 | 0,902 | 0,951 | 0,962 | 0,988 | 1,011 |
| $I_4$ | 1,112 | 1,135 | 1,165 | 0,859 | 0,908 | 0,946 | 0,959 | 0,979 | 1,029 |
| $I_5$ | 0,992 | 1,005 | 1,015 | 0,974 | 0,997 | 1,015 | 0,969 | 0,991 | 1,010 |
| $I_6$ | 0,956 | 0,998 | 1,024 | 0,920 | 0,942 | 0,985 | 0,914 | 0,945 | 0,990 |
| $I_7$ | 0,996 | 1,013 | 1,045 | 0,926 | 0,951 | 0,973 | 0,919 | 0,940 | 0,965 |
| $I_8$ | 0,985 | 1,003 | 1,024 | 0,974 | 1,007 | 1,046 | 0,977 | 1,003 | 1,033 |
| $I_9$ | 1,015 | 1,044 | 1,081 | 0,882 | 0,936 | 0,967 | 0,893 | 0,918 | 0,949 |
| $I_{10}$ | 0,908 | 0,953 | 0,991 | 0,905 | 0,931 | 0,954 | 0,980 | 0,994 | 1,017 |

TABLEAU 3.32 – Résultats pour la métrique DM des algorithmes évolutionnaires hybridés par MCE5

| Instance | NSGA2-MCE5 | | | SPEA2-MCE5 | | | MOAIS-MCE5 | | |
|---|---|---|---|---|---|---|---|---|---|
| Index | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 149,933 | 158,806 | 164,751 | 86,116 | 103,205 | 115,113 | 76,230 | 100,228 | 132,000 |
| $I_2$ | 132,831 | 155,161 | 164,900 | 72,780 | 84,832 | 108,503 | 61,830 | 77,385 | 88,114 |
| $I_3$ | 162,117 | 173,446 | 184,196 | 104,800 | 122,670 | 139,918 | 73,811 | 95,236 | 131,852 |
| $I_4$ | 264,940 | 275,966 | 292,306 | 102,835 | 130,038 | 148,064 | 107,224 | 143,762 | 173,784 |
| $I_5$ | 87,356 | 101,487 | 125,714 | 80,951 | 101,847 | 142,769 | 81,308 | 94,465 | 114,000 |
| $I_6$ | 37,000 | 100,311 | 141,028 | 37,216 | 89,069 | 130,115 | 63,055 | 114,892 | 257,853 |
| $I_7$ | 29,189 | 135,527 | 232,032 | 40,012 | 83,100 | 201,119 | 11,402 | 143,025 | 336,034 |
| $I_8$ | 110,132 | 127,306 | 155,910 | 98,899 | 126,444 | 163,997 | 92,190 | 117,014 | 149,609 |
| $I_9$ | 80,318 | 290,845 | 459,526 | 64,281 | 110,041 | 145,217 | 82,873 | 148,445 | 229,486 |
| $I_{10}$ | 87,863 | 353,864 | 522,188 | 0,000 | 21,634 | 68,066 | 33,015 | 150,702 | 273,308 |

TABLEAU 3.33 – Nombre de solutions non-dominées des les algorithmes évolutionnaires hybridés par MCE5

| Instance | NSGAII-MCE5 | | | SPEA2-MCE5 | | | MOAIS-MCE5 | | |
|---|---|---|---|---|---|---|---|---|---|
| Index | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 20 | 22,3 | 25 | 15 | 17,6 | 21 | 15 | 19,4 | 24 |
| $I_2$ | 20 | 22,1 | 24 | 13 | 14,7 | 16 | 14 | 15,8 | 18 |
| $I_3$ | 19 | 20,4 | 21 | 13 | 14,8 | 16 | 11 | 13,3 | 17 |
| $I_4$ | 24 | 27,9 | 32 | 16 | 18,4 | 21 | 18 | 21,9 | 26 |
| $I_5$ | 17 | 18,3 | 20 | 17 | 19,1 | 23 | 17 | 21,2 | 25 |
| $I_6$ | 4 | 6,3 | 9 | 5 | 7,8 | 11 | 9 | 11,6 | 15 |
| $I_7$ | 2 | 4,3 | 7 | 2 | 3,6 | 6 | 2 | 6,2 | 11 |
| $I_8$ | 12 | 15,7 | 18 | 12 | 17,4 | 20 | 15 | 17,9 | 22 |
| $I_9$ | 4 | 6,0 | 8 | 3 | 5,3 | 7 | 7 | 9,3 | 11 |
| $I_{10}$ | 3 | 6,1 | 9 | 1 | 2,7 | 4 | 2 | 5,0 | 8 |

Selon la métrique DM, les algorithmes évolutionnaires (voir tableau 3.32) ont des fronts de Pareto avec un étalement similaire sur les petites instances. Par contre, sur les grandes instances ($I_6$, $I_7$, $I_9$ et $I_{10}$), le classement des algorithmes par ordre décroissant est : NSGAII-MCE5, MOAIS-MCE5 et SPEA2-MCE5. Le nombre de solutions non-dominées dans le front de Pareto est équivalent pour chaque instance dans le groupe des algorithmes évolutionnaires hybridés, (voir tableau 3.33). Les solutions dans le front Pareto sont plus nombreuses sur les instances de petite taille. À noter que le NSGAII-MCE5 obtient le plus de solutions non-dominées sur les instances $I_1$, $I_2$, $I_3$, $I_4$ et $I_{10}$. Pour les instances $I_5$, $I_6$, $I_7$, $I_8$ et $I_9$, les solutions non-dominées fournies par le MOAIS-MCE5 sont plus nombreuses.

TABLEAU 3.34 – Résultats de la métrique C des MOACO-MCE5, MOPSO-MCE5 et MOABC-MCE5

| | MOACO/MOPSO | | | MOACO/MOABC | | | MOPSO/MOABC | | |
|---|---|---|---|---|---|---|---|---|---|
| Instance | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 0,158 | 0,377 | 0,545 | 0,000 | 0,045 | 0,118 | 0,000 | 0,044 | 0,167 |
| 7/3/10 | 0,000 | 0,306 | 0,500 | 0,667 | 0,835 | 1,000 | 0,421 | 0,583 | 0,706 |
| $I_2$ | 0,222 | 0,350 | 0,471 | 0,000 | 0,038 | 0,187 | 0,000 | 0,046 | 0,125 |
| 8/4/8 | 0,083 | 0,279 | 0,500 | 0,750 | 0,866 | 1,000 | 0,389 | 0,483 | 0,550 |
| $I_3$ | 0,187 | 0,364 | 0,588 | 0,000 | 0,015 | 0,083 | 0,000 | 0,057 | 0,250 |
| 10/3/10 | 0,143 | 0,321 | 0,500 | 0,500 | 0,767 | 1,000 | 0,375 | 0,480 | 0,588 |
| $I_4$ | 0,000 | 0,271 | 0,500 | 0,000 | 0,029 | 0,105 | 0,000 | 0,022 | 0,062 |
| 10/4/15 | 0,000 | 0,228 | 0,643 | 0,600 | 0,822 | 0,929 | 0,333 | 0,391 | 0,560 |
| $I_5$ | 0,300 | 0,488 | 0,750 | 0,000 | 0,022 | 0,048 | 0,000 | 0,023 | 0,091 |
| 10/5/10 | 0,154 | 0,349 | 0,684 | 0,727 | 0,885 | 1,000 | 0,789 | 0,905 | 1,000 |
| $I_6$ | 0,000 | 0,047 | 0,222 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 10/5/20 | 0,429 | 0,898 | 1,000 | 1,000 | 1,000 | 1,000 | 0,571 | 0,908 | 1,000 |
| $I_7$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 10/5/30 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 0,750 | 0,975 | 1,000 |
| $I_8$ | 0,000 | 0,036 | 0,143 | 0,000 | 0,000 | 0,000 | 0,000 | 0,005 | 0,048 |
| 15/5/15 | 0,600 | 0,846 | 1,000 | 1,000 | 1,000 | 1,000 | 0,833 | 0,948 | 1,000 |
| $I_9$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 15/5/30 | 0,750 | 0,908 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 |
| $I_{10}$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 15/5/45 | 0,500 | 0,917 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 |

TABLEAU 3.35 – Résultats de la métrique H des MOACO-MCE5, MOPSO-MCE5 et MOABC-MCE5

| Instance | MOACO/MOPSO | | | MOACO/MOABC | | | MOPSO/MOABC | | |
|---|---|---|---|---|---|---|---|---|---|
| Index | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 0,935 | 0,983 | 1,023 | 0,930 | 0,961 | 0,982 | 0,916 | 1,000 | 1,083 |
| $I_2$ | 0,871 | 0,940 | 0,985 | 0,953 | 0,973 | 0,997 | 0,886 | 0,945 | 0,988 |
| $I_3$ | 0,924 | 0,963 | 1,009 | 0,948 | 0,990 | 1,053 | 0,858 | 0,938 | 0,983 |
| $I_4$ | 0,912 | 0,976 | 1,029 | 0,898 | 0,930 | 0,963 | 0,939 | 0,992 | 1,044 |
| $I_5$ | 0,990 | 1,014 | 1,059 | 0,961 | 0,975 | 1,001 | 0,942 | 0,961 | 0,980 |
| $I_6$ | 0,917 | 0,943 | 0,991 | 0,839 | 0,880 | 0,905 | 0,878 | 0,913 | 0,942 |
| $I_7$ | 0,915 | 0,947 | 0,978 | 0,853 | 0,880 | 0,941 | 0,925 | 0,964 | 0,987 |
| $I_8$ | 0,919 | 0,954 | 0,984 | 0,860 | 0,882 | 0,904 | 0,898 | 0,926 | 0,958 |
| $I_9$ | 0,918 | 0,955 | 0,984 | 0,813 | 0,849 | 0,883 | 0,889 | 0,913 | 0,933 |
| $I_{10}$ | 0,914 | 0,947 | 0,999 | 0,801 | 0,847 | 0,890 | 0,931 | 0,951 | 0,967 |

Pour le deuxième groupe, le MOABC-MCE5 a de meilleurs résultats que les MOACO-MCE5 et MOPSO-MCE5, en particulier sur les grandes instances $I_5$, $I_6$, $I_7$, $I_8$, $I_9$ et $I_{10}$ (voir tableau 3.34). Le MOSPSO-MCE5 domine le MOACO-MCE5 sur les instances $I_6$, $I_7$, $I_8$, $I_9$ et $I_{10}$, par contre les deux algorithmes obtiennent des résultats équivalents sur les instances de petite taille $I_1$, $I_2$, $I_3$, $I_4$ et $I_5$.

TABLEAU 3.36 – Résultat pour la métrique DM des MOACO-MCE5, MOPSO-MCE5 et MOABC-MCE5

| Instance | MOACO-MCE5 | | | MOPSO-MCE5 | | | MOABC-MCE5 | | |
|---|---|---|---|---|---|---|---|---|---|
| Index | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 82,207 | 104,457 | 118,861 | 122,801 | 143,776 | 162,493 | 70,824 | 94,220 | 112,290 |
| $I_2$ | 57,957 | 85,280 | 110,313 | 125,128 | 142,022 | 154,716 | 59,657 | 76,181 | 87,476 |
| $I_3$ | 95,037 | 121,748 | 140,798 | 137,058 | 161,335 | 173,046 | 72,684 | 96,833 | 141,792 |
| $I_4$ | 129,430 | 147,066 | 176,912 | 218,794 | 242,397 | 268,293 | 93,739 | 127,759 | 168,57 |
| $I_5$ | 94,821 | 105,632 | 117,273 | 85,440 | 102,976 | 117,898 | 87,630 | 95,417 | 102,103 |
| $I_6$ | 51,546 | 95,318 | 146,762 | 32,542 | 64,010 | 95,901 | 42,697 | 61,422 | 80,405 |
| $I_7$ | 70,064 | 119,256 | 174,519 | 0,000 | 36,445 | 112,357 | 31,048 | 84,654 | 119,101 |
| $I_8$ | 80,225 | 118,218 | 150,569 | 94,372 | 125,757 | 182,217 | 97,693 | 129,188 | 173,899 |
| $I_9$ | 98,473 | 133,850 | 185,451 | 8,944 | 30,877 | 58,301 | 26,382 | 101,753 | 186,172 |
| $I_{10}$ | 158,266 | 240,617 | 381,063 | 0,000 | 24,929 | 89,196 | 56,134 | 144,830 | 262,120 |

TABLEAU 3.37 – Nombre de solutions non-dominées du second groupe hybridé par MCE5

| Instance | MOACO-MCE5 | | | MOPSO-MCE5 | | | MOABC-MCE5 | | |
|---|---|---|---|---|---|---|---|---|---|
| Index | Min | Moy | Max | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 9 | 11,8 | 14 | 16 | 18,8 | 22 | 16 | 18,0 | 21 |
| $I_2$ | 10 | 11,9 | 16 | 17 | 19,8 | 24 | 14 | 15,3 | 18 |
| $I_3$ | 11 | 13,1 | 14 | 15 | 16,4 | 18 | 11 | 12,6 | 16 |
| $I_4$ | 9 | 11,6 | 14 | 14 | 19,1 | 25 | 16 | 17,7 | 20 |
| $I_5$ | 10 | 13,1 | 19 | 16 | 19,3 | 22 | 20 | 22,0 | 25 |
| $I_6$ | 4 | 6,0 | 9 | 4 | 9,1 | 13 | 8 | 9,7 | 11 |
| $I_7$ | 2 | 3,8 | 5 | 1 | 2,9 | 5 | 2 | 4,3 | 7 |
| $I_8$ | 10 | 11,6 | 15 | 8 | 13,1 | 18 | 14 | 18,5 | 22 |
| $I_9$ | 4 | 4,8 | 6 | 2 | 3,4 | 5 | 4 | 5,4 | 7 |
| $I_{10}$ | 2 | 4,0 | 6 | 1 | 1,9 | 3 | 3 | 4,1 | 5 |

Pour le deuxième groupe, le MOACO obtient sa meilleure évaluation à la DM sur l'instance $I_{10}$ (voir tableau 3.37). Le MOPSO-MCE5 a des résultats très irréguliers selon la DM. Sur les petites instances ($I_1$, $I_2$, $I_3$ et $I_4$), il possède un bon étalement du front de Pareto. Avec les instances $I_5$ et $I_8$, le MOPSO-MCE5 a des résultats équivalents à la DM aux MOACO-MCE5 et MOABC-MCE5. Par contre le MOPSO-MCE5 a de très mauvais résultats selon la DM sur les grandes instances :

$I_6$, $I_7$, $I_9$ et $I_{10}$. Les résultats du MOABC-MCE5 à la DM sont bons et réguliers sur toutes les instances. Pour le nombre de solutions dans le front de Pareto, le MOPSO-MCE5 obtient plus de solutions non-dominées pour les instances de petites tailles $I_1$, $I_2$, $I_3$, et $I_4$ et le MOABC-MCE5 a plus d'individus dans le front de Pareto optimal sur les instances $I_5$, $I_6$, $I_7$, $I_8$, $I_9$ et $I_{10}$ (voir tableau 3.37). À noter que le nombre de solutions dans le front de Pareto est plus important pour les instances $I_1$, $I_2$, $I_3$, $I_4$, $I_5$ et $I_8$.

TABLEAU 3.38 – Résultats de la métrique C pour la comparaison des meilleurs de chaque groupe

| | MOAIS/MOABC | | | (MOAIS/MOABC)-MCE5 | | |
|---|---|---|---|---|---|---|
| Instance | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 0,050 | 0,152 | 0,263 | 0,059 | 0,179 | 0,412 |
| 7/3/10 | 0,550 | 0,725 | 0,900 | 0,529 | 0,665 | 0,737 |
| $I_2$ | 0,000 | 0,107 | 0,214 | 0,071 | 0,310 | 0,533 |
| 8/4/8 | 0,533 | 0,792 | 1,000 | 0,312 | 0,497 | 0,722 |
| $I_3$ | 0,000 | 0,106 | 0,231 | 0,000 | 0,202 | 0,312 |
| 10/3/10 | 0,588 | 0,817 | 1,000 | 0,467 | 0,650 | 0,846 |
| $I_4$ | 0,000 | 0,108 | 0,200 | 0,000 | 0,090 | 0,210 |
| 10/4/15 | 0,682 | 0,803 | 1,000 | 0,565 | 0,739 | 0,954 |
| $I_5$ | 0,000 | 0,023 | 0,045 | 0,000 | 0,115 | 0,273 |
| 10/5/10 | 0,750 | 0,908 | 1,000 | 0,545 | 0,743 | 0,909 |
| $I_6$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 10/5/20 | 0,733 | 0,885 | 1,000 | 0,692 | 0,883 | 1,000 |
| $I_7$ | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 10/5/30 | 0,800 | 0,960 | 1,000 | 0,667 | 0,927 | 1,000 |
| $I_8$ | 0,000 | 0,034 | 0,100 | 0,000 | 0,015 | 0,059 |
| 15/5/15 | 0,800 | 0,905 | 1,000 | 0,800 | 0,936 | 1,000 |
| $I_9$ | 0,000 | 0,027 | 0,167 | 0,000 | 0,033 | 0,167 |
| 15/5/30 | 0,500 | 0,723 | 0,875 | 0,556 | 0,792 | 1,000 |
| $I_{10}$ | 0,000 | 0,033 | 0,333 | 0,000 | 0,025 | 0,250 |
| 15/5/45 | 0,333 | 0,781 | 1,000 | 0,600 | 0,837 | 1,000 |

En termes de convergence et de diversité, les meilleurs algorithmes de chaque groupe sont le MOAIS et le MOABC pour les classiques, le MOAIS-MCE5 et le MOABC-MCE5 pour les algorithmes hybridés. Ces algorithmes sont comparés entre eux avec les métriques C et H (voir tableau 3.38 et 3.39). Les MOABC classiques et hybridés dominent réciproquement les MOAIS classiques et hybridés sur toutes les instances. Selon la DM, les MOABC classiques et hybridés ont un front de Pareto moins étendu que les MOAIS classiques et hybridés (voir tableaux 3.20, 3.24, 3.32 et 3.36). L'hybridation MCE5 réduit la largeur du front de Pareto des MOAIS et MOABC. Pour le nombre de solutions, les MOABC et MOABC-MCE5 ont généralement moins de solutions non-dominées

TABLEAU 3.39 – Résultats de la métrique H pour la comparaison des meilleurs de chaque groupe

| Instance | MOAIS/MOABC | | | (MOAIS/MOABC)-MCE5 | | |
|---|---|---|---|---|---|---|
| Index | Min | Moy | Max | Min | Moy | Max |
| $I_1$ | 0,935 | 0,975 | 1,011 | 0,979 | 0,991 | 1,011 |
| $I_2$ | 0,941 | 0,978 | 1,018 | 0,980 | 0,998 | 1,011 |
| $I_3$ | 0,890 | 0,960 | 1,018 | 0,965 | 0,990 | 1,036 |
| $I_4$ | 0,937 | 0,969 | 1,018 | 0,937 | 0,975 | 1,015 |
| $I_5$ | 0,926 | 0,949 | 0,972 | 0,967 | 0,986 | 1,005 |
| $I_6$ | 0,914 | 0,937 | 0,965 | 0,907 | 0,945 | 0,996 |
| $I_7$ | 0,914 | 0,948 | 0,980 | 0,945 | 0,963 | 0,984 |
| $I_8$ | 0,866 | 0,900 | 0,928 | 0,870 | 0,899 | 0,937 |
| $I_9$ | 0,928 | 0,950 | 0,983 | 0,892 | 0,947 | 1,007 |
| $I_{10}$ | 0,963 | 0,978 | 0,995 | 0,944 | 0,981 | 0,995 |

que les MOAIS et MOAIS-MCE5 (voir tableaux 3.21, 3.25, 3.33 et 3.37).

## 3.5 Conclusion

Dans ce chapitre, la MCE combinée avec le HC est adaptée aux problèmes multiobjectifs Np-difficiles. Le problème de référence est un FJSP biobjectif résolu par des métaheuristiques fréquemment rencontrées dans la littérature : NSGAII (Deb et al., 2002), SPEA2 (Zitzler et al., 2001), MOAIS (Yoo and Hajela, 1999), MOACO (Mariano et al., 1999), MOPSO (Coello Coello and Lechuga, 2002) et MOABC (Pham and Ghanbarzadeh, 2007). Plusieurs combinaisons d'hybridation sont testées sur le NSGAII, pour trouver la plus performante. Le fonctionnement de chaque hybridation est étudié en fonction du nombre de PRL et de LS sur les solutions non-dominées, dominées et créées dans les zones inexplorées. L'hybridation la plus performante est celle qui effectue des recherches locales sur les solutions des zones fortement explorées ($zfe$) et créées dans les zones inexplorées ($zi$), mais elle nécessite le plus de calculs. Les autres expérimentations fournissent un classement des méthodes multiobjectifs avec et sans hybridation. Il ressort que le MOABC est le plus performant des algorithmes avec et sans hybridation pour résoudre le FJSP multiobjectif. L'utilisation de la MCE avec la recherche locale améliore les performances de tous les algorithmes utilisés. Par contre, les algorithmes se comportent différemment avec la MCE, le NSGAII, le SPEA2 et le MOACO réagissent mieux à l'hybridation, tandis que le MOABC déjà le plus performant obtient des améliorations plus modestes.

# Conclusion générale

Cette thèse propose une étude méthodologique sur l'optimisation difficile. L'objectif est d'étudier l'exploration de l'espace de recherche par des méthodes d'optimisation. Une nouvelle hybridation se référant à l'espace des variables est proposée pour améliorer la convergence des méthodes d'optimisation. Dans le premier chapitre, un état de l'art présentant un bilan sur les méthodes d'optimisation pour les problèmes NP-difficiles. Les méthodes exactes donnent les résultats optimaux, mais sont mal adaptées aux instances de grandes tailles. À l'inverse, les méthodes approchées fournissent des résultats plus rapidement, mais ceux-ci sont souvent non-optimaux. Elles ont des fonctionnements et un nombre de paramètres à régler différents, mais elles doivent toutes trouvées le bon équilibre entre intensification et diversification. L'intensification fait converger l'algorithme vers les optimums globaux au fil des itérations. La diversification ralentit la convergence mais sort l'algorithme des minimums locaux rencontrés. Le réglage des paramètres de l'algorithme pour obtenir cet équilibre est déterminé par le programmeur. Les méthodes approchées dépendent donc largement de l'expérience de l'utilisateur. L'objectif de cette thèse est de fournir une hybridation permettant d'améliorer les performances de toutes les méta-heuristiques.

Les métaheuristiques évoluent dans l'espace des objectifs s'intéressant aux solutions avec les meilleurs notes obtenues à l'évaluation par la fonction objectif. L'approche proposée s'intéresse à l'espace des variables pour orienter la recherche des méthodes d'optimisation. Les solutions sont identifiées en fonction de leurs caractéristiques pour être ensuite utilisées par l'algorithme.

Dans le deuxième chapitre, une modélisation facilement manipulable est proposée pour l'espace de variable. Le principe de la MCE est simple, faire correspondre chaque représentation de solution à un et un seul numéro. Ces numéros sont classés sur un axe uni-dimensionnel par rapport à une solution de référence. La conversion d'une solution en un numéro se fait en trois étapes. La première convertit la solution avec des bits en décimal en vecteur binaire. Ce vecteur en binaire est comparé à celui de la solution de référence, pour obtenir une représentation de la distance de *Hamming*.

Cette représentation en binaire de la solution est ensuite convertie en un numéro avec la FCC qui est une fonction bijective.

Avec la FCC, les solutions avec le moins de différences par rapport à la solution de référence selon la distance de *Hamming* obtiennent les plus petits numéros, celles avec le plus de différences reçoivent les plus grands numéros. Sur cet espace uni-dimensionnel des zones sont créées, toutes les

solutions qui ont la même distance de *Hamming* se retrouvent dans la même zone.

Lorsqu'un algorithme explore l'espace de solutions, la répartition est évaluée en fonction des zones. La qualité de l'exploration de chaque zone est déterminée : $zi$ pour celles sans solutions explorées et $ze$ pour celles avec au moins une solution explorée. Les zones $ze$ sont requalifiées en $znd$ pour celles avec au moins l'une des meilleures solutions explorées et les $zfe$ pour celles avec un nombre de solutions explorées supérieur à la moyenne. Intuitivement pour guider l'exploration d'une métaheuristique, l'intensification de la recherche se fait à partir des solutions dans les $znd$ et $zfe$ et la diversification passe par la création de solutions dans les $zi$. L'exploration du voisinage des solutions est utilisée pour guider la recherche dans l'espace de solutions. Pour intensifier la recherche de la métaheuristique, un HC est appliqué sur les solutions dans les $znd$ ou $zfe$. En explorant le voisinage des solutions explorées, des solutions sont créées dans les $zi$. Un HC est ensuite appliqué sur ces nouvelles solutions.

Un problème d'ordonnancement, fréquent dans la littérature, est choisi comme problème de référence le FJSP. Le FSJP est un problème NP-difficile avec l'objectif de minimisation *makespan* (Garey et al., 1976). Il est résolu par de nombreux algorithmes tels que le GA (Pezzella et al., 2008), le AIS (Bagheri et al., 2010), l'ACO (Xing et al., 2010) et le PSO (Girish and Jawahar, 2009) pour ne citer qu'eux. Un GA pour le FSJP est utilisé comme algorithme de référence. Lors des expérimentations, le GA classique et plusieurs hybridations sont comparés. Le HC est utilisé comme recherche locale. Tous les algorithmes s'arrêtent après le même temps d'exécution écoulé. Les hybridations implémentées sont le GA avec LS sur les solutions non-dominées (GA-LS), le GA avec LS sur les solutions dans les $znd$ (GA-MCE1) et le GA avec LS sur les solutions dans les $znd$ et créées dans les $zi$ (GA-MCE4). Les expérimentations montrent que le GA-LS à une meilleure convergence que le GA. Mais les hybridations GA-MCE1 et GA-MCE4 convergent beaucoup plus vite que le GA-LS.

L'adaptation de la MCE pour les problèmes multiobjectifs est abordée dans le troisième chapitre. Le problème de référence est le FJSP avec les objectifs de minimisation du *makespan* et de la production des lots juste à temps. Les algorithmes pour résoudre le FJSP multiobjectif sont de la même nature que ceux pour le FJSP mono-objectif : les MOEA (Chiang and Lin, 2013), les MOAIS (Lu, 2009), les MOACO (Xing et al., 2009b), les MOPSO (Xia and Wu, 2005) et les MOABC (Wang et al., 2012a).

Le fonctionnement de la MCE pour les problèmes d'optimisation multiobjectifs est similaire à celui pour les problèmes mono-objectifs. Pour l'orientation, les zones avec les meilleures solutions explorées sont remplacées par les zones contenants des solutions non-dominées.

L'un des plus classiques MOEA, le NSGAII (Deb et al., 2002) est utilisé pour l'adaptation de la MCE. Cinq hybridations sont étudiées, appliquant la HC sur les solutions de différents types de zones : NSGAII-MCE1 pour les $znd$, NSGAII-MCE2 pour les $zfe$, NSGAII-MCE3 pour les $zi$, NSGAII-MCE4 pour les $znd$ et les $zi$ et NSGAII-MCE5 pour les $zfe$ et les $zi$. Ces hybridations

sont comparées aux NSGAII et NSGAII-LS qui appliquent une recherche locale sur les solutions non-dominées. Les résultats de la métrique C confirment que le NSGAII-LS domine le NSGAII. Le NSGAII-MCE2 est meilleur que les NSGAII-(MCE1,MCE3) car il intensifie davantage sa recherche que le NSGAII-MCE1 et surpasse la diversification du NSGAII-MCE3. Le NSGAII-MCE4 obtient de moins bons résultats que le NSGAII-MCE5 ; pour les mêmes raisons le NSGAII-MCE2 est plus performant que le NSGAII-MCE1. Selon la métrique-C, le NSGAII-LS est dominé par le NSGAII-(MCE2,MCE5), l'utilisation de la MCE combinée avec le HC permet d'améliorer les résultats d'un algorithme. Le NSGAII-MCE5 surclasse le NSGAII-MCE2 grâce à la diversification de sa recherche dans les $zi$.

Six métaheuristiques sont comparées entre elles avec et sans l'hybridation Alg-MCE5 (NSGAII (Deb et al., 2002), SPEA2 (Zitzler et al., 2001), MOAIS (Yoo and Hajela, 1999), MOACO (Mariano et al., 1999), MOPSO (Coello Coello and Lechuga, 2002) et MOABC (Pham and Ghanbarzadeh, 2007)). Les résultats montrent que l'hybridation améliore les performances de tous les algorithmes. Le MOAIS obtient de meilleurs résultats que le NSGAII et le SPEA2. Et le MOABC est le plus performant des six algorithmes avec et sans hybridation.

Cette thèse a introduit un nouveau concept : utiliser l'espace des variables pour guider l'exploration de l'espace de recherche par une métaheuristique. L'efficacité de cette approche est expérimentée sur un problème NP-difficile avec plusieurs métaheuristiques.

Malgré des résultats encourageants, plusieurs axes d'études restent à approfondir. Le choix de la solution de référence se fait aléatoirement. Ce choix aléatoire peut être remplacé par une méthode qui choisit la solution avec la meilleure influence sur la répartition des solutions dans l'espace uni-dimensionnel. Le classement des solutions sur la carte unidimensionnelle se fait, après une conversion en binaire, en fonction de la distance de *Hamming* par rapport à une solution de référence. Il serait intéressant de proposer une autre formule qui organise les solutions sur l'axe selon d'autres caractéristiques, par exemple classer les solutions en fonction du nombre de gènes différents par rapport à la solution de référence, pour éviter une conversion binaire consommatrice de temps d'exécution.

La recherche locale est utilisée avec la MCE, mais d'autres hybridations peuvent être utilisées. Tel qu'un contrôleur à logique floue ou un réseau de neurones qui règle dynamiquement la convergence et la diversité dans chaque zone. La coévolution peut aussi collaborer avec la MCE, par exemple en cherchant à construire des solutions dans des zones précises lors de l'assemblage des individus.

Le découpage fournit des zones de tailles irrégulières, certaines sont très grandes par rapport à d'autres. Un partitionnement en sous-zones est à faire pour les zones de grandes tailles. Le défaut le plus visible de la MCE est le temps d'exécution qu'il serait intéressant de réduire. La MCE est utilisée avec la recherche locale ; d'autres hybridations peuvent fournir une collaboration intéressante avec la MCE. Il reste à généraliser la MCE en l'appliquant à d'autres problèmes et/ou autres méthodes d'optimisation.

La MCE peut être aussi combinée avec des méthodes exactes, par exemple en utilisant la MCE pour découper l'espace de solutions en zones et une méthode exacte pour trouver les meilleures solutions dans chaque zone. Les solutions obtenues sont ensuite comparées pour obtenir les solutions optimales.

Selon le même principe, une méthode approchée peut remplacer la méthode exacte. Chaque zone est explorée par une méthode approchée. Les meilleures solutions de chaque zone sont ensuite comparées entre elles pour obtenir les meilleures solutions trouvées par l'algorithme.

Une collaboration entre méthodes exactes et approchées avec la MCE est envisageable. La MCE découpe l'espace de recherche en petites zones. L'exploration de l'espace de recherche se fait en alternant l'utilisation de méthode exacte et approchée. Une méthode approchée fournit ses meilleures solutions après avoir convergé. Ces solutions sont utilisées pour identifier des zones. Les meilleures solutions contenues par ces zones sont identifiées par une méthode exacte. Ces solutions sont ensuite ajoutées à la population avant de relancer la méthode approchée.

Cette thèse a donné lieu à plusieurs publications, trois communications de conférence internationale (Autuori et al., 2013b) (Autuori et al., 2012) (Autuori et al., 2014) et une communication de conférence nationale (Autuori et al., 2013a). Un article de revue internationale est également soumis.

# Références bibliographiques


H.A. Abbass, R. Sarker, and C. Newton. PDE : A pareto-frontier differential evolution approach for multi-objective optimization problems. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 2, pages 971–978, 2001.

A. Abel and P. Korhonen. Using aspiration levels in an interactive interior multiobjective linear programming algorithm. *European Journal of Operational Research*, 89(1) :193–201, 1996.

J. Adams, E. Balas, and D. Zawack. The shifting bottleneck procedure for job shop scheduling. *Management Science*, 34(3) :391–401, 1988.

R.B. Agrawal and K. Deb. Simulated binary crossover for continuous search space. Technical report, 1994.

R.T.F. Ah King, B. Radha, and H.C.S. Rughooputh. A fuzzy logic controlled genetic algorithm for optimal electrical distribution network reconfiguration. In *Networking, Sensing and Control, 2004 IEEE International Conference on*, volume 1, pages 577–582, 2004.

R. Akbari, R. Hedayatzadeh, K. Ziarati, and B. Hassanizadeh. A multi-objective artificial bee colony algorithm. *Swarm and Evolutionary Computation*, 2(0) :39–52, 2012.

J. Andersson and D. Wallace. Pareto optimization using the struggle genetic crowding algorithm. *Engineering Optimization*, 34(6) :623–643, 2002.

J. Autuori, F. Hnaien, F. Yalaoui, A. Hamzaoui, and Essounbouli N. Comparison of the solution space exploration by NSGAII and SPEA2 for the flexible job shop problem. 2012.

J. Autuori, F. Hnaien, and Yalaoui. Étude de l'exploration de l'espace de solution pour le problème du job-shop flexible bi-objectifs par des algorithmes génétiques. 2013a.

J. Autuori, F. Hnaien, F. Yalaoui, A. Hamzaoui, and Essounbouli N. Comparison of the solution space exploration by NSGAII and SPEA2 for the flexible job shop problem. 2013b.

J. Autuori, F. Hnaien, and F. Yalaoui. A guided exploration method of genetic algorithm for flexible job shop problem. 2014.

A. Bagheri, M. Zandieh, I. Mahdavi, and M. Yazdani. An artificial immune algorithm for the flexible job-shop scheduling problem. *Future Generation Computer Systems*, 26(4) :533–541, 2010.

J. Balicki, Z. Kitowski, and A. Stateczny. Extended hopfield models of neural networks for combinatorial multiobjective optimization problems. In *Neural Networks Proceedings, 1998. IEEE*

*World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on*, volume 2, pages 1646–1651, 1998.

S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb. A simulated annealing-based multiobjective optimization algorithm : AMOSA. *IEEE Transactions on Evolutionary Computation*, 12(3) : 269–283, 2008.

James C Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA journal on computing*, 6(2) :154–160, 1994.

R.P. Beausoleil. "MOSS" multiobjective scatter search applied to non-linear multiple criteria optimization. *European Journal of Operational Research*, 169(2) :426–449, 2006.

R.P. Beausoleil Delgado. Multiple criteria scatter search. *Proceedings of the 4th Metaheuristics International Conference (MIC'2001)*, pages 539–543, 2001.

J. Behnamian and S.M.T. Fatemi Ghomi. Development of a PSO-SA hybrid metaheuristic for a new comprehensive regression model to time-series forecasting. *Expert Systems with Applications*, 37(2) :974–984, 2010.

J. Bekker and C. Aldrich. The cross-entropy method in multi-objective optimisation : An assessment. *European Journal of Operational Research*, 211(1) :112–121, 2011.

R. Bellman and R. Kalaba. *Dynamic programming and modern control theory*. Academic Press New York, 1965.

C. Blum, J. Puchinger, G.R. Raidl, and A. Roli. Hybrid metaheuristics in combinatorial optimization : A survey. *Applied Soft Computing Journal*, 11(6) :4135–4151, 2011.

P. Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 41(3) :157–183, 1993.

C.S. Chang, D.Y. Xu, and H.B. Quek. Pareto-optimal set based multiobjective tuning of fuzzy automatic train operation for mass transit system. *Electric Power Applications, IEE Proceedings -*, 146(5) :577–583, 1999.

R. Chelouah and P. Siarry. A hybrid method combining continuous tabu search and nelder-mead simplex algorithms for the global optimization of multiminima functions. *European Journal of Operational Research*, 161(3) :636–654, 2005.

T.C. Chiang and H.J. Lin. A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling. *International Journal of Production Economics*, 141(1) :87–98, 2013.

L. Climent, M.A Salido, and F. Barber. Robustness in dynamic constraint satisfaction problems. *International Journal of Innovative Computing Information and Control*, 8(4) :2513–2532, 2012.

C.A. Coello Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms : a survey of the state of the art. *Computer methods in applied mechanics and engineering*, 191(11) :1245–1287, 2002.

C.A. Coello Coello and G.B. Lamont. *Applications of multi-objective evolutionary algorithms*, volume 1. World Scientific, 2004.

C.A. Coello Coello and M.S. Lechuga. MOPSO : a proposal for multiple objective particle swarm optimization. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 2, pages 1051–1056, 2002.

C.A. Coello Coello and G.T. Pulido. Multiobjective structural optimization using a microgenetic algorithm. *Structural and Multidisciplinary Optimization*, 30 :388–403, 2005.

A. Colorni, M. Dorigo, and M. Vittorio. Distributed optimization by ant colonies. In *Proceedings of the first European conference on artificial life*, pages 134–142, 1992.

D.W. Corne, J.D. Knowles, and M.J. Oates. The pareto envelope-based selection algorithm for multiobjective optimization. In *Parallel Problem Solving from Nature PPSN VI*, pages 839–848, 2000.

G.B Dantzig and RW.E. Cottle. Maximization of a linear function of variables subject to linear inequalities. *The Basic*, pages 24–32, 2003.

G.B. Dantzig and M.N. Thapa. Linear programming 2 : Theory and extensions. 2003.

S. Dauzère-Pérès and J. Paulli. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research*, 70 :281–306, 1997.

L.N. De Castro and F.J. Von Zuben. Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation*, 6(3) :239–251, 2002.

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, 2002.

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.

M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *Computational Intelligence Magazine, IEEE*, 1(4) :28–39, 2006.

J.D. Farmer, N.H. Packard, and A.S. Perelson. The immune system, adaptation, and machine learning. *Physica D : Nonlinear Phenomena*, 22(1-3) :187 – 204, 1986.

T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6 :109–133, 1995.

H. Fisher and G.L. Thompson. Probabilistic learning combinations of local job-shop scheduling rules. *Industrial scheduling*, 3 :225–251, 1963.

J. Fliege, L.M.G. Drummond, and B.F. Svaiter. Newton's method for multiobjective optimization. *SIAM Journal on Optimization*, 20(2) :602–626, 2009.

D.B. Fogel. Evolutionary algorithms in theory and practice. *Complexity*, 2(4) :26–27, 1997.

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

M.R. Garey, D.S. Johnson, and R. Sethi. The complexity of flowshop and job shop scheduling. *Mathematics of Operations Research*, 1 :117–129, 1976.

Z.W. Geem, J.H. Kim, and G.V. Loganathan. A new heuristic optimization algorithm : Harmony search. *Simulation*, 76(2) :60–68, 2001.

B.S. Girish and N. Jawahar. A particle swarm optimization algorithm for flexible job shop scheduling problem. In *Automation Science and Engineering, 2009. CASE 2009. IEEE International Conference on*, pages 298–303, 2009.

F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8 (1) :156–166, 1977. ISSN 1540-5915.

F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5) :533–549, 1986.

F. Glover. A template for scatter search and path relinking. In *Artificial Evolution*, volume 1363, pages 1–51. Springer Berlin Heidelberg, 1998.

F. Glover, E. Taillard, and E. Taillard. A user's guide to tabu search. *Annals of Operations Research*, 41(1) :1–28, 1993.

F. Glover, M. Laguna, and R. MartÃŋ. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 29(3) :652–684, 2000.

J.F. Goncalves, J.J. De Magalhaes Mendes, and M.G.C. Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, 167(1) :77–95, 2005.

P. Greistorfer. A tabu scatter search metaheuristic for the arc routing problem. *Computers and Industrial Engineering*, 44(2) :249–266, 2003.

Y.V. Haimes, L.S. Lasdon, and D.A. Wismer. 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, 1971.

M.P. Hansen. Tabu search for multiobjective optimization : MOTS. *Presented at MCDM 97*, 1997.

M.P. Hansen. Tabu search for multiobjective combinatorial optimization : TAMOCO. *Control and Cybernetics*, 29(3) :799–818, 2000.

P. Hansen. The steepest ascent mildest descent heuristic for combinatorial programming. In *Proceedings of the Congress on Numerical Methods in Combinatorial Optimization*, 1986.

P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2) :100–107, 1968.

A. Hertz, B. Jaumard, C. Ribeiro, and W. F. Filho. A multi-criteria tabu search approach to cell formation problems in group technology with multiple objectives. *RAIRO/Operations Research*, 28(3) :302–328, 1994.

H.H. Hoos and T. Stützle. *Stochastic local search : Foundations & applications*. Elsevier, 2004.

J. Horn, N. Nafpliotis, and D.E. Goldberg. Niched pareto genetic algorithm for multiobjective optimization. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, volume 1, pages 82–87, 1994.

R.-H. Huang, C.-L. Yang, and W.-C. Cheng. Flexible job shop scheduling with due window - a two-pheromone ant colony approach. *International Journal of Production Economics*, 141(2) : 685–697, 2013.

J. Hurink, B. Jurisch, and M. Thole. Tabu search for the job-shop scheduling problem with multi-purpose machines. *OR Spektrum*, 15(4) :205–215, 1994.

N. Imanipour and S.H. Zegordi. A heuristic approach based on tabu search for early/tardy flexible job shop problems. *Scientia Iranica*, 13(1) :1–13, 2006.

M. Jiangming, K. Hirasawa, H. Jinglu, and J. Murata. Genetic symbiosis algorithm for multiobjective optimization problem. In *Robot and Human Interactive Communication, 2000. RO-MAN 2000. Proceedings. 9th IEEE International Workshop on*, pages 137–142, 2000.

I. Kacem, S. Hammadi, and P. Borne. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man and Cybernetics Part C : Applications and Reviews*, 32(1) :1–13, 2002a.

I. Kacem, S. Hammadi, and P. Borne. Pareto-optimality approach for flexible job-shop scheduling problems : Hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and Computers in Simulation*, 60(3-5) :245–276, 2002b.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *Encyclopedia of Machine Learning*, volume 4, pages 1942–1948, 1995.

M.S. Kiran, M. Gunduz, and O.K. Baykan. A novel hybrid algorithm based on particle swarm and ant colony optimization for finding the global minimum. *Applied Mathematics and Computation*, 219(4) :1515–1521, 2012.

J. Knowles and D. Corne. The pareto archived evolution strategy : a new baseline algorithm for pareto multiobjective optimisation. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 1, pages 3 vol. (xxxvii+2348), 1999.

F. Kursawe. A variant of evolution strategies for vector optimization. In *Parallel Problem Solving from Nature*, pages 193–197. Springer Berlin / Heidelberg, 1991.

L. Kyung-Mi, T. Yamakawa, and L. Keon-Myung. A genetic algorithm for general machine scheduling problems. In *Knowledge-Based Intelligent Electronic Systems, 1998. Proceedings KES '98. 1998 Second International Conference on*, volume 2, pages 60–66, 1998.

M. Lan, T. Xu, and L. Peng. Solving flexible multi-objective JSP problem using a improved genetic algorithm. *Journal of Software*, 5(10) :1107–1113, 2010.

A.H. Land and A.G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3) :497–520, 1960.

H.C.W. Lau, T.M. Chan, W.T. Tsui, F.T.S. Chan, G.T.S. Ho, and K.L. Choy. A fuzzy guided multi-objective evolutionary algorithm model for solving transportation problem. *Expert Systems with applications*, 36 :8255–8266, 2009.

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

J.-Q. Li, Q.-K. Pan, and Y.-C. Liang. An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. *Computers and Industrial Engineering*, 59(4) :647–662, 2010.

J.-Q. Li, Q.-K. Pan, and K.-Z. Gao. Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems. *International Journal of Advanced Manufacturing Technology*, 55(9-12) :1159–1169, 2011a.

J.-Q. Li, Q.-K. Pan, P.N. Suganthan, and T.J. Chua. A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, 52(5-8) :683–697, 2011b.

J.-Q. Li, S.-X. Xie, Q.-K. Pan, and S. Wang. A hybrid artificial bee colony algorithm for flexible job shop scheduling problems. *International Journal of Computers, Communications and Control*, 6(2) :286–296, 2011c.

L.Z. Liao and D. Li. Adaptive differential dynamic programming for multiobjective optimal control. *Automatica*, 38(6) :1003–1015, 2002.

Q. Lin and J. Chen. A novel micro-population immune multiobjective optimization algorithm. *Computers and Operations Research*, 2011.

H. Liu, A. Abraham, and Z. Wang. A multi-swarm approach to multi-objective flexible job-shop scheduling problems. *Fundamenta Informaticae*, 95(4) :465–489, 2009.

H. Lu. Stretching technique-based clonal selection algorithm for flexible job-shop scheduling. Number 2, pages 111–114, 2009.

C. Mariano and E. Morales. A new approach for the solution of multiple objective optimization problems based on reinforcement learning. In *MICAI 2000 : Advances in Artificial Intelligence*, volume 1793, pages 212–223. Springer, 2000a.

C. Mariano and E. Morales. A new distributed reinforcement learning algorithm for multiple objective optimization problems. In *Advances in Artificial Intelligence*, volume 1952, pages 290–299. Springer, 2000b.

C. Mariano, E. Morales, C.E. Mariano, P. Cuauhnahuac, and E. Morales. A multiple objective ant-Q algorithm for the design of water distribution irrigation networks. Technical report, 1999.

C.E. Mariano and E.F. Morales. Distributed reinforcement learning for multiple objective optimization problems. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, volume 1, pages 188–195, 2000c.

A. Martelli. An application of heuristic search methods to edge and contour detection. *Communications of the ACM*, 19(2) :73–83, 1976.

S.Z. Martinez, A.A. Montano, and C.A.C. Coello. A nonlinear simplex search approach for multi-objective optimization. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 2367–2374, 2011.

L. D. Merkle. A random function based framework for evolutionary algorithms. In *Proceedings of the 7th International Conference on Genetic Algorithms. 105-112*, pages 105–112. Morgan Kauffman, 1997.

U. Montanari. Networks of constraints : Fundamental properties and applications to picture processing. *Information sciences*, 7 :95–132, 1974.

P. Moscato et al. On evolution, search, optimization, genetic algorithms and martial arts : Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, 826, 1989.

G. Moslehi and M. Mahnam. A pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *International Journal of Production Economics*, 129(1) :14–22, 2011.

S. Mostaghim and J. Teich. Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO). In *Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE*, pages 26–33, 2003.

A.N.K. Nasir and M.O. Tokhi. Novel metaheuristic hybrid spiral-dynamic bacteria-chemotaxis algorithms for global optimisation. *Applied Soft Computing Journal*, 27 :357–375, 2015.

J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4) :308–313, 1965.

T. Niknam. A new HBMO algorithm for multiobjective daily volt/var control in distribution systems considering distributed generators. *Applied Energy*, 88(3) :778–788, 2011.

Q.-K. Pan, L. Wang, and B. Qian. A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems. *Computers and Operations Research*, 36(8) :2498–2511, 2009.

J. Paredis. Coevolutionary algorithms. *Evolutionary computation*, 2 :224–238, 1998.

K.E. Parsopoulos and M.N. Vrahatis. Particle swarm optimization method in multiobjective problems. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 603–607, 2002.

L. Perelman and A. Ostfeld. Multi-objective design of water distribution systems using cross entropy. In *Impacts of Global Climate Change*, page 35, 2005.

F. Pezzella, G. Morganti, and G. Ciaschetti. A genetic algorithm for the flexible job-shop scheduling problem. *Computers and Operations Research*, 35(10) :3202–3212, 2008.

D.T. Pham and A. Ghanbarzadeh. Multi-objective optimisation using the bees algorithm. *IPROMS2007 - 2007 Innovative Production machines and systems*, 2007.

D.T. Pham, A. Ghanbarzadeh, E. KoÃğ, S. Otri, S. Rahim, and Zaidi M. The bees algorithm, a novel tool for complex optimization problems. *Intelligent Production Machines and Systems*, pages 454–459, 2006.

M. A. Potter and K. A. De Jong. A cooperative coevolutionary approach to function optimization. pages 249–257, 1994.

A. Przybylski, X. Gandibleux, and M. Ehrgott. Two phase algorithms for the bi-objective assignment problem. *European Journal of Operational Research*, 185(2) :509–533, 2008.

S. Qu, M. Goh, and F.T.S. Chan. Quasi-newton methods for solving multiobjective optimization. *Operations Research Letters*, 39(5) :397–399, 2011.

S.H.A. Rahmati, M. Zandieh, and M. Yazdani. Developing two multi-objective evolutionary algorithms for the multi-objective flexible job shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, 64(5-8) :915–932, 2013.

S. Ramakrishnan and Y.A. Hasan. Fuzzy preference-based multi-objective optimization method. *Artificial Intelligence Review*, 39(2) :165–181, 2013.

T. Ray and K.M. Liew. A swarm metaphor for multiobjective design optimization. *Engineering Optimization*, 34(2) :141–153, 2002.

C.R. Reeves, editor. *Modern heuristic techniques for combinatorial problems.* John Wiley & Sons, Inc., 1993.

A.P. Reynolds, D.W. Corne, and B. de la Iglesia. A multiobjective GRASP for rule selection. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 643–650, 2009.

J. Ricart, G. HÃijttemann, J. Lima, and B. BarÃąn. Multiobjective harmony search algorithm proposals. *Electronic Notes in Theoretical Computer Science*, 281 :51–67, 2011.

E. Rollon and J. Larros. Constraint optimization techniques for exact multi-objective optimization. *Lecture Notes in Economics and Mathematical Systems*, 618 :89–98, 2009.

S. Ruan, W. Fan, Z. Gao, J. Lin, R. Xin, and Z. Zhao. Non-dominated sorting simplex algorithm for multi-objective optimization. In *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on*, volume 2, pages 457–461, 2010.

R. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1 :127–190, 1999.

R.Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1) :89–112, 1997.

X. Shao, W. Liu, Q. Liu, and C. Zhang. Hybrid discrete particle swarm optimization for multi-objective flexible job-shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, 67(9-12) :2885–2901, 2013.

N.Z. Shor. Cut-off method with space extension in convex programming problems. *Cybernetics*, 13 :94–96, 1977.

B.S. Stewart and C.C. White III. Multiobjective A. *Journal of the ACM*, 38(4) :775–814, 1991.

R. Storn and K. Price. Differential evolution : A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11 :341–359, 1997.

T. Stutzle and H.H. Hoos. MAX-MIN ant system. *Future Generation Computer Systems*, 16(8) : 889–914, 2000.

S.M. Tabatabaei and B. Vahidi. Bacterial foraging solution based fuzzy logic decision for optimal capacitor allocation in radial distribution system. *Electric Power Systems Research*, 81(4) :1045–1050, 2011.

D.W. Tank and J.J. Hopfield. Simple 'neural' optimization networks : An A/D converter, signal decision circuit, and a linear programming circuit. *IEEE transactions on circuits and systems*, CAS-33(5) :533–541, 1986.

J.C. Tay and D. Wibowo. An effective chromosome representation for evolving flexible job shop schedules. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3103 :210–221, 2004.

E.L. Ulungu and J. Teghem. The two phases method : An efficient procedure to solve biobjective combinatorial optimization problems. *Foundation of computing and decision science*, 20 :149–156, 1995.

E.L. Ulungu, J. Teghem, P.H. Fortemps, and D. Tuyttens. MOSA method : a tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 8(4) :221–236, 1999.

A. Unveren and A. Acan. Multi-objective optimization with cross entropy method : Stochastic learning with clustered pareto fronts. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 3065–3071, 2007.

D.A. Van Veldhuizen. Multiobjective evolutionary algorithms : classifications, analyses, and new innovations. Technical report, 1999.

D.A. Van Veldhuizen and G.B. Lamont. Multiobjective optimization with messy genetic algorithms. In *Proceedings of the 2000 ACM symposium on Applied computing*, volume 1, pages 470–476, 2000.

R.J. Vanderbei, M.S. Meketon, and B.A. Freedman. A modification of karmarkar's linear programming algorithm. *Algorithmica*, 1(1-4) :395–407, 1986.

M. Visée, J. Teghem, M. Pirlot, and E.L. Ulungu. Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. *Journal of Global Optimization*, 12(2) : 139–155, 1998.

L. Wang, G. Zhou, Y. Xu, and M. Liu. An enhanced pareto-based artificial bee colony algorithm for the multi-objective flexible job-shop scheduling. *International Journal of Advanced Manufacturing Technology*, 60(9-12) :1111–1123, 2012a.

L. Wang, G. Zhou, Y. Xu, S. Wang, and M. Liu. An effective artificial bee colony algorithm for the flexible job-shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, 60(1-4) :303–315, 2012b.

C.J.C.H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4) :279–292, 1992.

G. Weil, K. Heus, P. Francois, and M. Poujade. Constraint programming for nurse scheduling. *Engineering in Medicine and Biology Magazine, IEEE*, 14(4) :417–422, 1995.

W. Weng and S. Fujimura. Distributed-intelligence approaches for weighted just-in-time production. *IEEE Transactions on Electrical and Electronic Engineering*, 5(5) :560–568, 2010.

Z. Wu and M.X. Weng. Multiagent scheduling method with earliness and tardiness objectives in flexible job shops. *IEEE Transactions on Systems, Man, and Cybernetics, Part B : Cybernetics*, 35(2) :293–301, 2005.

W. Xia and Z. Wu. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers and Industrial Engineering*, 48(2) :409–425, 2005.

L.-N. Xing, Y.-W. Chen, and K.-W. Yang. Double layer ACO algorithm for the multi-objective fjssp. *New Generation Computing*, 26(4) :313–327, 2008.

L.-N. Xing, Y.-W. Chen, and K.-W. Yang. An efficient search method for multi-objective flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, 20(3) :283–293, 2009a.

L.-N. Xing, Y.-W. Chen, and K.-W. Yang. Multi-objective flexible job shop schedule : Design and evaluation by simulation modeling. *Applied Soft Computing Journal*, 9(1) :362–376, 2009b.

L.-N. Xing, Y.-W. Chen, P. Wang, Q.-S. Zhao, and J. Xiong. A knowledge-based ant colony optimization for flexible job shop scheduling problems. *Applied Soft Computing Journal*, 10(3) : 888–896, 2010.

J. Yang, J. Zhou, L. Liu, and Y. Li. A novel strategy of pareto-optimal solution searching in multi-objective particle swarm optimization (MOPSO). *Computers and Mathematics with Applications*, 57(11-12) :1995–2000, 2009.

J. Yaochu, M. Olhofer, and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *Evolutionary Computation, IEEE Transactions on*, 6(5) :481–494, 2002.

P.-Y. Yin, S.-S. Yu, P.-P. Wang, and Y.-T. Wang. Multi-objective task allocation in distributed computing systems by hybrid particle swarm optimization. *Applied Mathematics and Computation*, 184(2) :407–420, 2007.

J. Yoo and P. Hajela. Immune network simulations in multicriterion design. *Structural and Multidisciplinary Optimization*, 18 :85–94, 1999.

L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3) :338–353, 1965.

G. Zhang, X. Shao, P. Li, and L. Gao. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering*, 56 : 1309–1318, 2009.

Q. Zhang and H. Li. MOEA/D : A multiobjective evolutionary algorithm based on decomposition. *Evolutionary Computation, IEEE Transactions on*, 11(6) :712–731, 2007.

G. Zhou, L. Wang, Y. Xu, and S. Wang. An effective artificial bee colony algorithm for multi-objective flexible job-shop scheduling problem. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6839 LNAI : 1–8, 2011.

E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms : A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4) :257–271, 1999.

E. Zitzler, M. Laumanns, and L. Thiele. SPEA2 : Improving the strength pareto evolutionnary algorithm. *Expert systems with applications*, 2001.

E. Ziztler, L. Thiele, M. Laumanns, C.M. Fonseca, and V. Grunert da Fonseca. Performance assessment of multiobjective optimizers. *IEEE Transactions on Evolutionary Computation*, 284 : 117–132, 2003.

# Julien AUTUORI

## Doctorat : Optimisation et Sûreté des Systèmes

Année 2014


## Energie, coopération méta-heuristiques et logique floue pour l'optimisation difficile


Au cours de cette thèse, l'exploration de l'espace de solutions par des métaheuristiques est abordée. Les métaheuristiques sont des méthodes d'optimisation utilisées pour résoudre des problèmes NP-difficiles. Elles explorent aléatoirement l'espace de recherche pour trouver les meilleures solutions. Dans un premier temps, l'ensemble des solutions est modélisé par un espace unidimensionnel par une Méthode de Conversion de l'Espace de recherche (MCE). Des métriques sont proposées pour évaluer l'exploration de l'espace de recherche par une métaheuristique en identifiant les zones explorées et inexplorées. Ces métriques sont utilisées pour orienter l'exploration de l'espace de recherche d'une méthode d'optimisation. La convergence est améliorée en accentuant le recherche dans les zones explorées. Pour sortir des minimums locaux, l'exploration est diversifiée en la dirigeant vers les zones inexplorées. En associant l'exploration du voisinage des solutions et ces métriques cartographiques, il est possible d'améliorer les performances des métaheuristiques. Plusieurs algorithmes mono-objectifs et multiobjectifs sont implémentés en version classique, hybridé par la recherche locale et par la MCE. Le Flexible Job Shop Problem (FJSP) est utilisé comme problème de référence. Les expérimentations avec les algorithmes hybridés montrent une amélioration des performances.



Mots clés : optimisation combinatoire - algorithmes d'approximation – ordonnancement (gestion).


## Energy, Cooperation Meta-heuristics and Fuzzy Logic for NP-hard Optimization


In this thesis, the solution space exploration by the metaheuristic is developed. The metaheuristics optimization methods are used to solve NP-hard problems. They explore randomly the search space to look for the best solutions. In a first step, the solution set is modeled by a one-dimensional space by a Mapping Method (MaM). Metrics are proposed to evaluate the search space exploration by a metaheuristic, identifying the explored and unexplored zones. These metrics are used to guide the search space exploration of an optimization method. The convergence is improved by emphasizing the research in the zones explored. To get out local minima, the exploration is diversified by pointing it towards the unexplored zones. Combining the neighbour discovery of the solutions and these mapping metrics, it is possible to improve the performance of metaheuristics. Several single-objective and multi-objective algorithms are implemented in the classic version, hybridized with local search and MaM. The Flexible Job Shop Problem (FJSP) is used as a reference problem. The experimentations with hybridized algorithms show performance improved.



Keywords: combinatorial optimization - approximation algorithms – production scheduling.



**Thèse réalisée en partenariat entre :**

Le Grand Troyes

Université de Reims Champagne-Ardenne

utt université de technologie Troyes

**Ecole Doctorale** "Sciences et Technologies"