Projet de fin d'étude
informatique

Université Lille 1

Méthode de résolution pour le problème de planification des tâches multi-objectif

INRIA - Lille Nord Europe

Auteur:
Emilie Allart

Tuteurs:
Sophie Jacquin
Laetitia Jourdan

27 janvier 2016

Remerciements

Je remercie Laetitia Jourdan de m'avoir acceptée pour ce projet, c'est toujours un plaisir de travailler avec elle, merci pour ses conseils. Mais je tiens aussi à dire un grand merci à Sophie Jacquin qui m'a encadrée tout au long de ce projet, ce fut une bonne expérience de travailler avec elle.

Contents

Introduction

Dans le cadre de ma dernière année de master, j'ai effectué mon projet de fin d'étude à Inria Lille Nord Europe dans l'équipe Dolphin, afin de mettre en place une nouvelle méthode de résolution pour le problème de planification des tâches multi-objectif, encadrée par Sophie Jacquin (INRIA) et Laetitia Jourdan (INRIA/CRIStAL). Je vais donc dans un premier temps, présenter le contexte, puis poser le problème, pour ensuite expliquer les méthodes de résolution et l'application au problème, et finir par le protocole et l'analyse des résultats.

Inria est un établissement public de recherche à caractère scientifique et technologique. Il a été créé en 1967 et a pour mission de produire une recherche d'excellence dans les champs informatiques et mathématiques des sciences du numérique et de garantir l'impact de cette recherche. Il couvre l'ensemble du spectre des recherches au coeur de ces domaines d'activités, et intervient sur les questions en lien avec le numérique, posées par les autres sciences et par les acteurs économiques et sociétaux. Inria rassemble 1677 chercheurs de l'institut et 1772 universitaires ou chercheurs d'autres organismes, il compte plus de 4500 articles publiés en 2013 et est à l'origine de plus de 110 start-ups. L'institut est organisé en 8 centres : Bordeaux, Grenoble, Lille, Nancy, Rennes, Rocquencourt, Saclay et Sophia-Antipolis.

Inria Lille - Nord Europe comporte 16 équipes de recherche et possède plusieurs partenariats tels que Lille1, Lille2, Lille3, Centrale Lille, le CNRS et le CWI. La stratégie du centre est de développer autour de la métropole lilloise un pôle d'excellence de rayonnement international (en priorité vers l'Europe du nord) et à fort impact local. Pour se faire, l'institut s'appuie sur des thématiques de recherche ambitieuses dans le domaine des sciences du numérique; l'intelligence des données et les systèmes logiciels adaptatifs, plus précisément :

- Internet des données et Internet des objets
- Couplage perception/action pour l'interaction homme-machine
- Modèle patient personnalisé dynamique
- Génie logiciel pour les systèmes éternels

L'équipe Dolphin (Discrete multi-objective Optimization for Large-scale Problems with Hybrid dIstributed techNiques) entretient plusieurs relations industrielles et internationales (EDF-GDF, bioinformatique, DHL, Univ. Montréal, ...) De nombreux secteurs de l'industrie sont concernés par des problèmes d'optimisation à grande échelle et complexes mettant en jeux des coûts financiers très importants et pour lesquels les décisions doivent être prises de façon optimales. Face à des applications qui nécessitent la résolution de problèmes de taille sans cesse croissante et ce dans des délais de plus en plus court, voire en temps réel, seule la mise en oeuvre conjointe des méthodes avancées issues de l'optimisation combinatoire en Recherche Opérationnelle, de la décision en IA

et de l'utilisation du Parallélisme et de la distribution permettrait d'aboutir à des solutions satisfaisantes.

L'équipe Dolphin a pour objectif la modélisation et la résolution parallèle de problèmes d'optimisation combinatoire (multi-objectifs) de grandes tailles. Des méthodes parallèles coopératives efficaces sont développées à partir de l'analyse de la structure du problème traité. Les problèmes ciblés appartiennent aussi bien à la classe des problèmes génériques (ordonnancement flow-shop, élaboration de tournées, etc...) que des problèmes industriels issue de la logistique, du transport, de l'énergie et de la bioinformatique.

Le problème de planification des tâches (Job Shop Scheduling Problem) consiste à planifier le traitement d'un certain nombre de tâches par les machines d'un l'atelier. L'objectif le plus couramment etudié est de trouver le planning qui permette d'achever l'ensemble des tâches au plus tôt. Néanmoins, avec une politique du juste à temps, il est nécessaire de considérer simultanément, comme second critère, le respect maximal de dates d'échéance afin d'éviter les retards de livraison et les coûts de stockage. L'équipe Dolphin développe de nouvelles méthodes d'optimisation combinatoire multi-objectif pour ce problème, en particulier des métaheuristiques. Pour tester la qualité des méthodes proposées, il nous faut nous comparer aux méthodes existantes.

Chapter 1

Position du problème

Avant de présenter le problème en détail, posons les bases en introduisant l'optimisation multi-objectif.

1.1 Optimisation multi-objectif

L'optimisation multi-objectif permet de résoudre des problèmes d'optimisation présentant plus d'un objectif. La formule ci-dessous représente un problème de minimisation multi-objectif.

$$\underset{x\in\Omega}{Min}\ (f_1(x), f_2(x), ...f_M(x)) \tag{1.1}$$

Ω est l'espace des solutions réalisables, x est une solution, $f_i(x)$ est la $i^{ième}$ fonction objectif et M donne le nombre d'objectifs. Dans la plupart des cas, ces objectifs sont conflictuels, ce qui signifie que l'optimisation d'un objectif entraine la détérioration d'autres. De ce fait, on ne peut pas trouver une solution meilleure pour tous les objectifs simultanément, c'est pourquoi nous utilisons le principe d'optimalité Pareto.

Le principe d'optimalité Pareto pour l'optimisation mutli-objectif est basé sur la relation de dominance Pareto. En supposant que les fonctions objectifs sont à minimiser, une solution x est dite dominante par rapport à une solution y si $\forall i \in 1, ..., M\ f_i(x) \leq f_i(y)$ et $\exists i \in 1, ..., M\ f_i(x) < f_i(y)$. Cela correspond aux points rouges sur le graphe 1.1. Une solution x^* est Pareto optimale si elle n'est dominée par aucune solution de l'espace de solution. L'ensemble des solutions Pareto optimales est appelé l'ensemble Pareto optimal, et l'ensemble des vecteurs objectifs correspondant représente le front Pareto. Le but de la résolution de problèmes d'optimisation multi-objectifs (MOPs) est donc de trouver l'ensemble Pareto optimal.

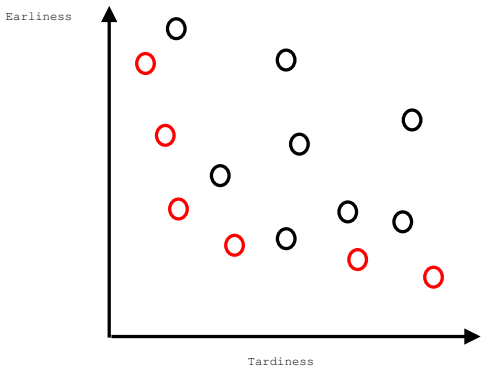

Figure 1.1: Solution pareto dominante

1.2 JobShop

Enoncé du problème

La planification de tâches (ou *job shop scheduling*) est un problème NP-complet. Son objectif est de minimiser l'écart entre les dates de complétion des tâches et leurs dates d'échéance.

Cependant, il est très difficile de trouver un planning respectant exactement les contraintes de retard et d'avance, donc notre objectif est de les minimiser au mieux. On considère donc deux objectifs : minimiser le retard et minimiser l'avance.

Le retard est la différence entre la complitude de la tâche i, notée C_i, c'est à dire le temps à laquelle elle est achevée, et la *due date*, notée d_i, le temps à laquelle elle aurait dû être fini, le tout pondéré par le facteur β spécifique à cette tâche. Et inversement pour l'avance avec un facteur de pondération α. Les facteurs α et β sont spécifiques à chaque tâche car il est plus ou moins important selon la tâche de la finir dans les temps impartis. Cependant, il faut prendre garde à respecter la disponibilité r de la ressource, une tâche ne peut pas être effectuée avant son temps r.

Modélisation mathématiques

N : le nombre de tâches

p_i : le temps de d'exécution de la tâche i

r_i : la disponibilité de tâche

I_i : le temps de pause d'une tâche i

$x_{i,j}$: variable binaire valant 1 si la tâche i est effectuée avant la tâche j.

Les fonctions objectifs à optimiser sont donc :

$$objectif1 : earliness = \sum_{i=0}^{N} \alpha_i \max((d_i - C_i), 0) \tag{1.2}$$

$$objectif2 : tardiness = \sum_{i=0}^{N} \beta_i \max((C_i - d_i), 0) \tag{1.3}$$

Il faut prendre garde à ce que les tâches ne se chevauchent pas, pour ce faire :

$$x_{i,j} + x_{j,i} = 1 \; \forall i, j \;\; et \;\; C_i <= C_j - p_j \; si \; x_{i,j} = 1 \tag{1.4}$$

De même, il faut respecter la disponibilité des tâches :

$$r_i <= C_i - p_i \; \forall i \tag{1.5}$$

Chapter 2

Méthodes de résolution

2.1 Principe générale

2.1.1 Algorithme évolutionnaire

Les algorithmes évolutionnaires sont inspirés du concept de sélection naturelle élaboré par Charles Darwin, d'ailleurs le vocabulaire employé découle de cette théorie.

- La fonction objectif F est appelée fonction de fitness
- Les points de l'espace de recherche Ω (pour notre cas se sont des ordonnancements) sont appelés des individus (ou chromosomes)
- L'ensemble des individus est une population
- Un tour de boucle principale de l'algorithme correspond à une génération

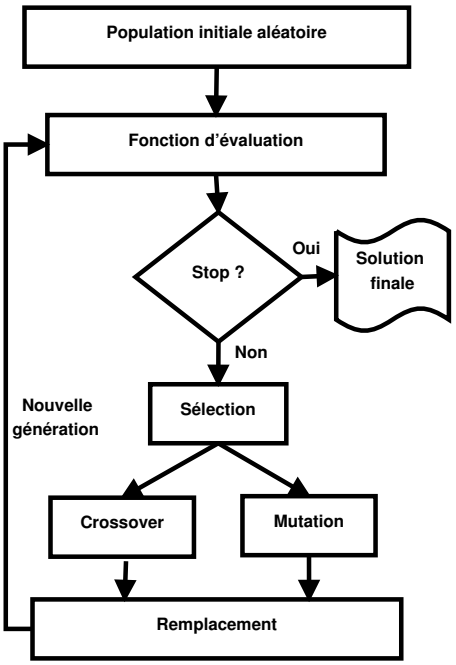

Figure 2.1: Schéma de AE

L'espace de recherche est déterminé par les contraintes du problème.
Le code ci-dessous 2.1.1 présente le principe général de l'algorithme.

```
population initiale
Tant que non(critere arret) :
  selection une partie de la population
  crossover des individus selectionnes
  mutation de la descendance
  evaluation de la fitness de chaque individu
  remplacement par une nouvelle population
```

Initialisation
L'initialisation consiste à échantilloner le plus uniformément possible l'espace de recherche Ω. Pour le cas de ce projet, on génère aléatoirement des ordonnancements de job.

Avec l'aide d'opérateurs génétiques, de nouveaux individus sont créés à partir des individus sélectionnés au préalable. On distingue les opérateurs de croisement (ou crossover) et les opérateurs de mutation. Ces transformations sont stochastiques, leur application nécessite des tirages aléatoires.

Mutation
La mutation a pour but d'apporter une modification légère aléatoire à une solution. Elle est appliquée avec une probabilité P_m sur chaque individu. Elle doit respecter la contrainte d'ergodicité, c'est-à-dire parcourir l'ensemble de recherche. Les détails des opérateurs choisis pour ce projet sont donnés dans le chapitre suivant.

Croisement
Cet opérateur reflète la "transmission d'information génétique" des parents. Si deux chromosomes parents sont performants, on peut considérer le fait qu'un de leur enfant pourrait hériter des parties qui les rendent performants et être ainsi meilleurs encore. Le croisement est appliqué avec une certaine probabilité P_c, ainsi il est possible de conserver certains chromosomes optimaux. On est dans le cadre d'une intensification (on garde les traits importants d'un individu). Il existe également une grande quantité d'opérateurs de croisement, ceux utilisés pour ce projet seront expliqués un peu plus loin.

Sélection
Le concept de Darwin s'intègre sous la forme de deux étapes dans l'algorithme : la sélection et le remplacement. La sélection consiste à sélectionner les indivudus à partir desquels on va générer la population enfant (sur lesquels on va appliquer les opérateurs de mutation et de crossover). Elle peut se faire de différentes façons :

- Le tirage de roulette consiste à donner à chaque individu une probabilité d'être sélectionné proportionnelle à sa performance.

- La sélection par le rang consiste à faire une sélection en utilisant une roulette dont les portions sont proportionnelles au rang des individus.
- La sélection par tournoi consiste à tirer T individus uniformément dans la population et à sélectionner le meilleur.

Remplacement
Le remplacement est l'étape où à partir de la population parent et de la population enfant, on constitue la génération suivante (de taille N). Il existe deux types de procédures, la sélection déterministe lors de laquelle on sélectionne les meilleurs individus selon la fitness. Les individus moins performants sont totalement éliminés, seuls les meilleurs restent, on parle d'élitisme. Et la sélection stochastique, le but est là aussi de sélectionner les meilleurs individus, mais de manière stochastique. Le meilleur n'est pas choisi à tous les coups.

Là aussi il y a plusieurs stratégies, par exemple:

- Remplacement générationnel total: la population enfant remplace totalement la population .
- Remplacement à élitisme total : on prend les N meilleurs individus parmis la population parent et l'enfant.
- Remplacement à élitisme faible : on prend l'enfant pour la prochaine génération, mais on remplace les k pire individus des enfants par les k meilleurs des parents

Lors des stratégies de remplacement et de sélection, on a besoin de classer les individus. La relation de Pareto équivalence n'est pas suffisante puisque certaines solutions seront équivalentes.
Les métaheuristiques se basent donc sur l'attribution de valeur de convergence et de diversification pour évaluer les solutions. La façon dont ces critères sont attribués permet de caractériser les algorithmes évolutionnaires. Une définition plus détaillée de ceux-ci :

Attribution d'une valeur de convergence : Le rôle de ce processus est de privilégier les solutions proches du front Pareto. Il doit permettre d'assurer la convergence de l'ensemble solution fourni par l'algorithme vers le front Pareto. Cette mesure permet donc de classer les individus de la population en fonction de leur proximité au front Pareto optimal. Il existe deux familles de méthode d'assignation de fitness. D'une part, les méthodes basées sur la relation de dominance et d'autre part les méthodes basées sur l'utilisation d'un indicateur de qualité. La relation de dominance a été décrite via la dominance Pareto. Les méthodes basées sur des indicateurs de qualité quant à elles utilisent un indicateur de qualité pour formuler le but de l'optimisation. Ainsi, le choix de l'indicateur représente l'objectif général du processus de recherche et les solutions de la population sont évaluées en fonction de cet indicateur.

Attribution d'une valeur de diversification : Cette stratégie a pour but d'éviter la convergence de la population vers un petit sous-ensemble de

solutions du front Pareto et donc évaluer chaque individu en fonction de l'impact qu'il a sur le critère de diversification de la population.

Plusieurs algorithmes évolutionnaires ont été proposés au cours du temps, on peut distinguer ceux utilisant directement le principe de dominance pour la procédure d'assignation de fitness et ceux utilisant des indicateurs de qualité.

2.2 NSGA II

Les algorithmes utilisant le principe de dominance sont ceux utilisant une procédure d'assignation de fitness basée sur la relation de dominance. Cette fitness permet de classer les individus en fonction de leur proximité avec le front Pareto mais elle ne garantit pas la construction d'une approximation du front ayant une bonne diversification. C'est pourquoi les individus se voient également assigner une mesure de diversification. Ainsi les individus sont classés en considérant dans un premier temps leur fitness puis leur mesure de diversification. Voici, par exemple, la description du NSGA II [8] (Non-dominated Sorting Genetic Algorithm II)

1. Mesure de convergence : Profondeur de dominance. Dans cette stratégie les individus sont regroupés par front Pareto de différents rangs. C'est le rang du front Pareto auquel il appartient qui permet d'évaluer un individu.

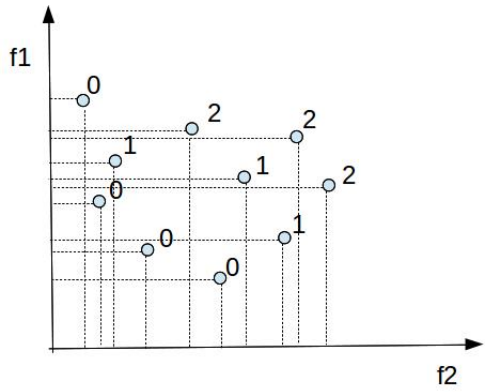

Figure 2.2: Profondeur de dominance

2. Mesure de diversité : Plus proche voisin en utilisant la crowding distance. La crowding distance fait la somme de la distance du point de son rang le plus proche pour chaque objectif.

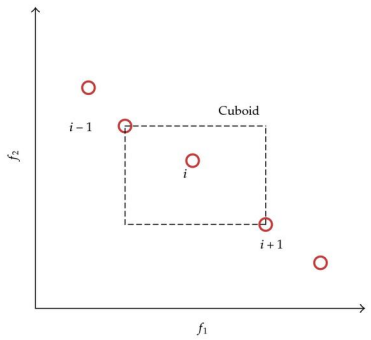

Figure 2.3: Crowding distance

3. Élitisme : Lors de l'étape de remplacement les individus parents et enfants sont classés selon leurs fitness, puis à fitness égale selon la valeur de leur apport en diversification, les meilleurs individus de ce classement sont ceux gardés pour constituer la génération suivante.

.

2.3 IBEA

Plus récemment des méthodologies basées sur l'utilisation d'un ou plusieurs indicateurs de qualité permettant d'évaluer l'apport d'une solution d'un point de vue convergence et/ou diversification ont été proposées. Ces méthodes n'utilisent pas forcément les deux processus d'assignation car l'indicateur de performance peut être choisi par rapport au critère que l'on souhaite évaluer (convergence et/ou diversification). Parmis ces algorithmes nous comptons l'IBEA [11] (Indicator-Based Evolutionary Algorithm) définit comme suit :

1. Mesure de convergence : Soit un indicateur binaire I, ici l'hypervolume différence, la valeur convergence se calcule comme suit : $convergence(x) = \sum_{y \in P} -e^{\frac{I(y,x)}{k}}$ où k est un réel à fixer. Cette fonction mesure la perte en qualité si la solution x est retirée de la population P.

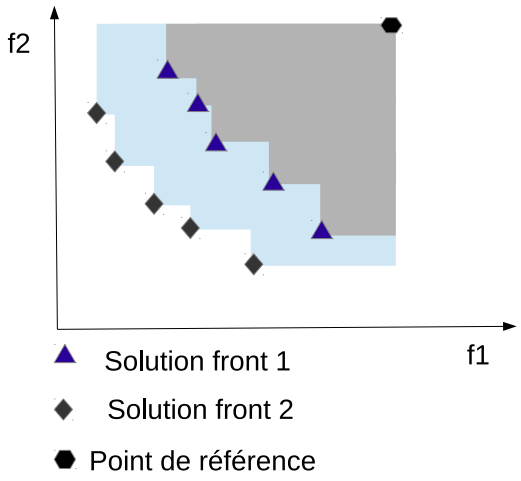

Figure 2.4: hypervolume

2. Mesure de diversité : Aucune.
3. Elitisme : Remplacement élitiste.

Chapter 3

Application au problème

3.1 Etat de l'art

3.1.1 Choix des opérateurs de mutation

Après avoir étudié la littérature, nous avons découvert l'article de *S. Kedad-Sidhoum* et *F. Sourd le roi* dans le domaine du Job Shop. [5] Leur modélisation est la seule qui intègre le concept de temps de pause (*idle time*) pour résoudre notre problème bi-objectif. En effet, il est possible d'avoir une pause entre deux tâches pour répondre au mieux aux objectifs.

Ainsi, une solution n'est pas représentée comme une suite de tâches mais comme une succession de blocs. Un bloc correspond à un ensemble de tâches successives ne nécessitant pas de pause. Chaque bloc, possède un temps de démarrage s avec une sous-séquence de tâches ordonnées et adjacentes après s. Cet algorithme permet premièrement, de faire un ensemble de changements au sein d'un bloc. Et deuxièmement, il utilise un opérateur temporel, qui permet de bouger en avant ou en arrière un bloc.
Les tâches d'un ordonnancement π sont partitionnées dans des blocs qui sont des séquences maximales de tâches ordonnancées sans pause. Posons $b(\pi)$ le nombre de blocs. De ce fait, les blocs d'un ordonnancement donné sont notés B_1, B_2, ..., B_b. $|B|$ correspond au nombre de jobs dans le bloc B et ${J_i}^B$,...,${J_{|B|}}^B$ identifient le job de B indéxé dans l'ordre de π.

Swap

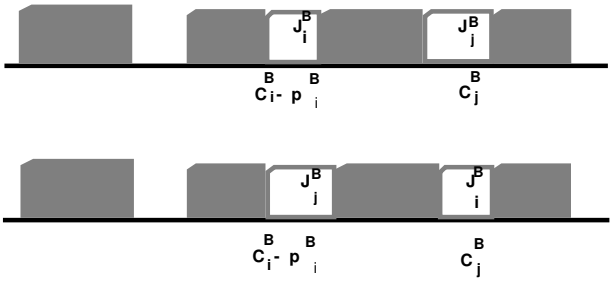

Figure 3.1: Swap de ${J_i}^B$ et ${J_j}^B$

Avec un ordonnancement π, la mutation *swap* consiste à échanger la position de deux jobs ${J_i}^B$ et ${J_j}^B$ d'un même bloc.

Extract-and-reinsert
Pour un ordonnancement π, le mutation consiste à enlever un job J de sa place dans π et en l'insérant devant ou derrière dans le même bloc. Les tâches du bloc sont décalées pour pouvoir insérer ce job à la position souhaitée. Pour un job ${J_i}^B$ d'un bloc B de π et une position $i \leq j \leq |B|$ telle que $i \neq j$, notons $[v_{ij}^B$ l'ordonnancement dérivé de π en décalant ${J_i}^B$ à une position derrière j dans B.

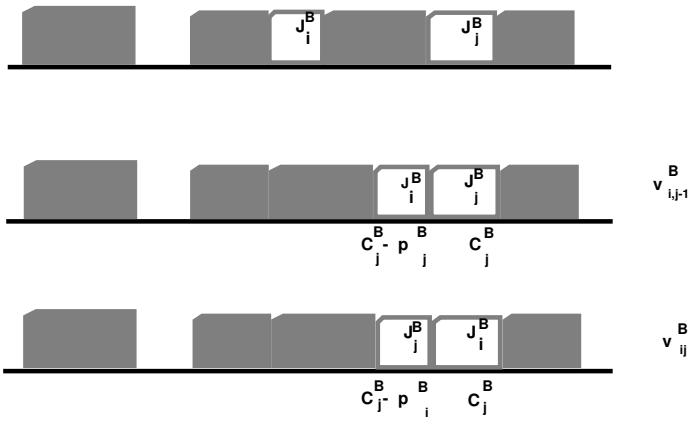

Figure 3.2: Extrait et reinsert ${J_i}^B$ à la position j-1 et j

subblock shift
Swap et extract-and-reinsert, permettent de modifier l'ordonnancement à l'intérieur d'un bloc. Subblock shift lui est un opérateur temporel, il permet de décaler les blocs en avant ou en arrière en réduisant les temps de pause, tout en conservant une solution réalisable. Son mécanisme est de sélectionner un job aléatoirement ${J_i}^B$. Si ce job se trouve en début de bloc, on a la possibilité de réduire son temps de pause I_B et ainsi fusionner avec le bloc de devant si le décalage $t = I_B$. Ou d'augmenter I_B en réduisant I_{B+1} pour ne pas modifier l'ordonnancement du bloc suivant et de fusionner avec le bloc suivant si $t = I_{B+1}$.

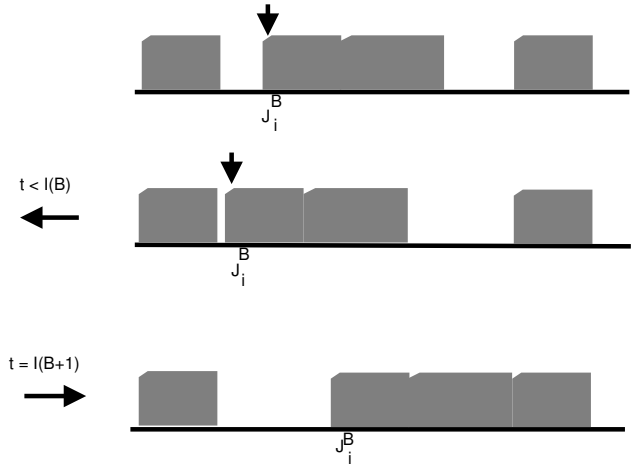

Figure 3.3: Décalage du bloc B

Dans les autres cas, le bloc est divisé en deux sous-blocs en intégrant un temps de pause t au sous-bloc de droite avac $t \leq I_{B+1}$, ce nouveau bloc peut éventuellement fusionner avec le bloc suivant.

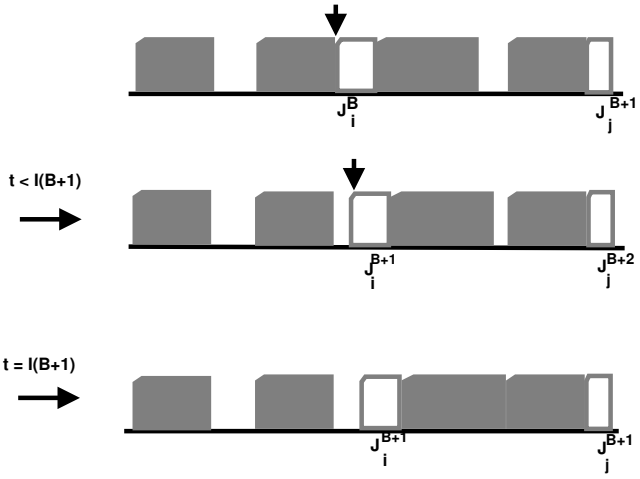

Figure 3.4: Décalage d'un sous-bloc de B

3.1.2 Choix des opérateurs de crossover

Après analyse de la littérature, nous avons comparé plusieurs opérateurs proposés et en avons relevé deux : le *2-point* et le *mask*. D'autres opérateurs ont été rencontrés notamment LOX (*Linear Order Crossover*), CX(*Cycle Crossover*) et PMX(*Partially Mapped Crossover*) qui sont comparés (*LOX* est supérieur au 2 autres) et expliqués dans [7]. Mais aussi *SJOX*, *RRX*, *BOUX* dans l'article [1].

2-point
Hybrid genetic algo with dominance prop Comme le montre l'article [10],qui compare plusieurs opérateurs de crossover avec son algorithme développé, le crossover est de loin celui qui a les meilleurs résultats. Le fait de garder l'ordre des parents aurait son importance. Le *2-point* présente plusieurs versions comme celle de [2], mais nous nous intéresserons à celle décrite dans l'article [3]. L'article [3] appuie l'efficacité de celui-ci.

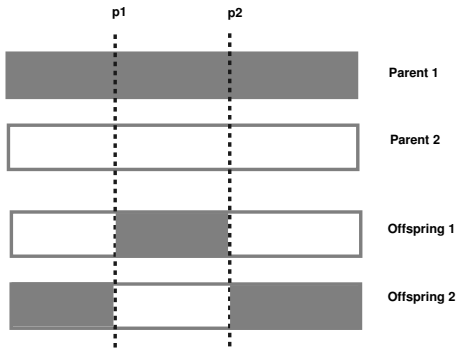

Figure 3.5: Déroulement du crossover 2-point

Son principe est simple, on tire aléatoirement de points $p1$ et $p2$. Puis comme le montre le schéma 3.5, on recopie la portion du $parent1$ dans l'$enfant1$ au même emplacement. Et on complète les tâches manquantes dans leur ordred'apparation dans le $parent2$ et inversement pour l'$enfant2$. Il existe une version légèrement différente qui copie à l'extérieur des points $p1$ et $p2$ puis complète l'intérieur.

Masque

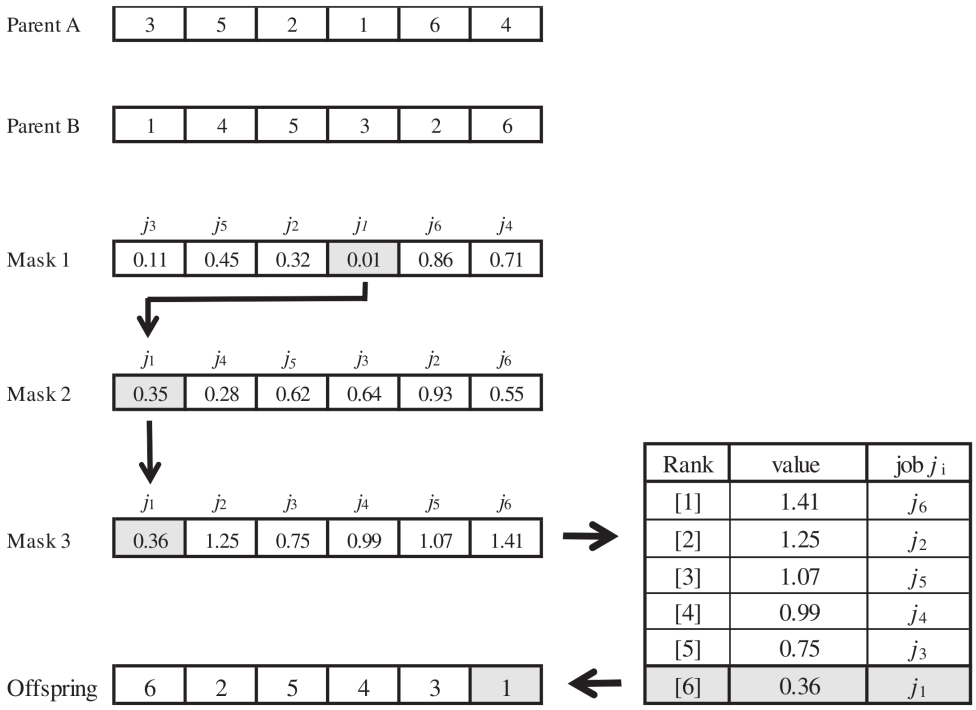

Figure 3.6: Déroulement du crossover Mask

Mis en avant par l'article *A genetic algorithm for JIT single machine scheduling with preemption and machine idle time*[6] , voici l'opérateur de crossover Mask. Pour commencer on génère deux nouveaux chromosomes *mask1* et *mask2* avec pour chaque gène une valeur de distribution aléatoire comprise entre $[0, 1]$ et un nombre de gènes égal à N. Comme nous pouvons le voir sur la figure 3.6, le *mask1* est associé au *parent*1 et le deuxième au *parent*2. Finalement, le *mask3* correspond à la somme des valeurs des gènes qui correspondent au même nombre dans le *mask1* et le *mask2*. Dans notre exemple, le gène associé au nombre 1 dans le *mask1* vaut 0.01 et celui associé à ce même nombre dans les *mask2* vaut 0.35. Donc le premier gène du *mask3*, qui est associé au nombre 1, est égal à la somme de ces deux valeurs donc 0.36. Ensuite, le *mask3* est trié selon la valeur de ses gènes, donnant en sortie le nombre qui lui est associé. Le nombre avec la plus grande valeur associée dans le *mask3* a la priorité pour se placer dans l'enfant. Ici, le nombre 1 est en $6^{ème}$ position.

3.2 Implémentation

Toutes les méthodes présentées ont été implémentées à l'aide du logiciel Paradiseo[4]. Paradiseo est un logiciel open-source qui fournit des frameworks permettant d'implémenter facilement des métaheuristiques.

3.2.1 Modélisation

On représente donc nos génotypes sous la forme d'un ensemble de blocs. Comme expliqué précédemment chaque bloc contient une succession de jobs ordonnés. Chaque job contient un temps de pause (nul pour tous sauf le premier de chaque bloc). Pour optimiser les accès aux données à nos opérateurs, nous avons présentés les données sous la forme d'une liste de paires associant un indice menant à une donnée dans la data et le temps de pause. Ainsi qu'une liste conservant dans l'ordre des blocs les indices dans le table précédente faisant référence à la première tâche du bloc.

Chapter 4

Protocole

4.1 Instances

Les instances utilisées sont issues d'une étude de F. Sourd [5]. Ces jeux d'instances sont répartis dans des fichiers dépendant du nombre d'instances dans celui-ci qui peut être de 20, 30, 40, 60, 89, 100, 130, 170 et 200. Dans notre étude, nous nous appuirons sur un fichier de taille 20 et un fichier de taille 60. Comme il a été présenté dans la présentation du problème, chaque instance contient p_i, d_i, r_i, α_i et β_i.

4.2 Mesure de performance

Pour notre étude, nous avons décidé d'utiliser l'ϵ-indicateur et l'hypervolume-différence, car ils sont complémentaires. Ceci avec Kruskall-Wallis car il est impossible de fixer le seed de chaque tests.

4.3 Paramétrage

  - La probabilité pour un individu de subir une mutation (choisie aléatoirement entre *swap*, *extract-and-reinsert* et *subblock*) : valeur testée dans l'intervalle [0, 1] avec une précision de $10^2 La probabilité pour un individu de subir un crossover 2\text{-}point$: $valeur testée dans l'intervalle [0, 1] avec une précision de 10^2$

- La probabilité pour un individu de subir un crossover *Mask*: valeur testée dans l'intervalle [0, 1] avec une précision de $10^2 Le choix de l'algorithme utilisé$: *NSGA II ou IBEA*

  Le critère d'arrêt est fixé en nombre de générations à 500. Et la taille de la population est fixée à N la taille de l'instance.

4.4 Mise en oeuvre des tests expérimentaux

Afin de comparer les méthodes, chacune d'entre elle est exécutée 20 fois sur chaque instance. Une analyse de sensibilité est effectuée pour fixer la valeur des

paramètres associés à chacune des méthodologies pour les différentes instances. Les tests statistique que nous effectuons sont des tests de Kruskall-Wallis, qui permettent de déterminer si une différence statistiquement significative existe entre les résultats fournis par les méthodes, et sont suivis, si la différence est significative, de tests post-hocs qui permettent de comparer les méthodes deux à deux afin de déterminer si l'une est meilleure que l'autre. Dans tous les cas une p-value de $0,01$ est utilisée.
Ces tests sont réalisés grâce au logiciel Pisa[9], qui permet de comparer statistiquement des échantillons de solutions de problème multi-objectif par rapport à un indicateur de performance donné.

Chapter 5

Résultats et Discussion

Les tableaux 5.1 et 5.2 présentent les résultats des comparaisons statistiques pour les différentes instances (de taille 20 et 60) en utilisant un indicateur d'hypervolume différence et l'ϵ-indicateur. Dans ces tableaux = indique une équivalence statistique entre les deux méthodes. Dans les autres cas, ils indiquent la valeur que doit prendre le paramètre, ou à partir de laquelle il doit prendre sa valeur, pour être meilleur. De plus, BestC représente l'opérateur de croisement le plus efficace avec les paramétrages précédents.

-

	2-point		Mask		BestC		Mutation	
	eps	hyp	eps	hyp	eps	hyp	eps	hyp
20	$\geq$0.5	=	$\geq$0.3	=	=	=	=	$\geq$ 0.3
60	=	=	=	=	=	=	=	=

Table 5.1: Tableau comparatif pour NSGA

Les résultats de l'algorithme $NSGAII$ montrent une influence de l'opérateur de crossover $2-point$ au delà de 0.5 et de l'opérateur $Mask$ au-delà de 0.3 sur le phénotype pour une instance de taille 20. Par contre les deux sont statistiquement équivalents comme on peut le voir avec la valeur de $BestC$, c'est pourquoi pour la suite de ces paramétrages nous avons utilisé le $2-point$ avec une probabilité de 0.6. De même la mutation pour toute probabilité ≤ 0.3 est statistiquement meilleure pour les instances de taille 20. Pour les instances de taille 60 par contre, les paramètres sont tous statistiquement équivalents.

	2-point		Mask		BestC		Mutation	
	eps	hyp	eps	hyp	eps	hyp	eps	hyp
20	=	=	=	=	=	=	=	=
60	=	=	=	=	=	Mask	=	=

Table 5.2: Tableau Comparatif pour $IBEA$

Les résultats obtenus pour l'algorithme $IBEA$ ne permettent pas de tirer beaucoup de conclusion. Les paramétrages sont tous statistiquements équivalents. Mis à part le fait que le $Mask$ est bien meilleur statistiquement pour des instances de taille 60.

Nous avons ensuite fait la comparaison du meilleur paramétrage de chaque algorithme. Pour des instances de taille 20, NSGA est statistiquement bien

meilleur . Et pour les instances de taille 60, il n'y a pas de différence, les algorithmes sont statistiquement équivalents. Il est donc préférable d'utiliser le NSGA avec une probabilité de crossover > 0.5

Conclusion

Les résultats obtenus ne nous ont pas permis de tirer de conclusions intéressantes, il faudrait utiliser les algorithmes implémentés et faire une étude en modifiant le nombre de générations ou prendre de plus grandes instances. Il n'a surement pas le temps de converger et reste donc trop éloigné du front.

Ce projet m'a permis de lire efficacement des papiers, de savoir choisir ce qui est le mieux adapter pour un problème donné et expliquer pourquoi. Même si ce travail n'a pas donné de résultats concluant pour l'instant, ce n'est pas perdu car les opérateurs développés ainsi que la nouvelle structure de données vont être intégrés à Paradiseo.

Bibliography

[1] Amrit Pratap Sameer Agarwal and T. Meyarivan. An adaptive genetic algorithm for solving the single machine scheduling problem with earliness and tardiness penalties. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 6, 2002.

[2] Chang P Chen S Fan C. A hybrid electromagnetism-like algorithm for single machine scheduling problem. *Expert Systems with Applications*, 2009.

[3] Murata T. Ishibuchi H. Performance evaluation of genetic algorithms for flowshop scheduling problems. *Evolutionary Multi-Criterion Optimization*, 2:812 – 817, 1994.

[4] A. Liefooghe L. Jourdan and E-G. Talbi. A software framework based on a conceptual unified model for evolutionary multiobjective optimization : Paradiseo-moeo. *European Journal of Operational Research*, 209(2):104–112, 2011.

[5] Francis Sourd Safia Kedad-Sidhoum. Earliness-tardiness scheduling with distinct due dates.

[6] H. Khorshidiana N. Javadiana M. Zandiehb J. Rezaeiana K. Rahmania. A genetic algorithm for jit single machine scheduling with preemption and machine idle time. *Expert Systems with Applications*, vol 38:7911–7918, 2011.

[7] Mahnam M Moslehi G Fatemi Ghomi S. Single machine scheduling with unequal release times and idle insert for minimizing the sum of maximum earliness and tardiness. *Mathematical and Computer Modelling*, 2013.

[8] Ribeiro F Souza S. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Expert Systems with Applications*, 2009.

[9] S. Bleuler M. Laumanns L. Thiele and E. Zitzler. Pisa : A platform and pro- gramming language independent interface for search algorithms. *Evolutionary Multi-Criterion Optimization*, pages 494–508, 2003.

[10] T. Starkweather S. McDaniel K. Mathias D. Whitley. A comparison of genetic sequencing operators.

[11] Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. *Swiss Federal Institute of Technology Zurich Computer Engineering and Networks Laboratory (TIK)*.

Glossaire

INRIA Institut National de Recherche en Informatique et en Automatique

CRIStal Centre de Recherche en Informatique, Signal et Automatique de Lille

CNRS Centre National de la Recherche Scientifique

CWI Centrum Wiskunde Informatica, organisme de recherche d'Amsterdam