École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Département Informatique

Cahier de spécification système & plan de développement			
Projet :	**Résolution d'un problème d'ordonnancement avec travaux interférants**		
Emetteur :	Baptiste Mille	Coordonnées : EPU-DI	
Date d'émission :	26 janvier 2014		
Validation			
Nom	Date	Valide (O/N)	Commentaires

Historique des modifications		
Version	Date	Description de la modification

00 : 03/01/2013 ; Version initiale

Table des matières

Cahier de spécification système

1.1 Introduction

Ce document constitue les spécifications système concernant le PFE [1] : "Méthodes approchées pour la résolution d'un problème d'ordonnancement avec travaux interférants".

Il présente et explicite, dans un premier temps, les différentes caractéristiques du projet : contexte, objectif, environnement et fonctionnalités. Puis dans un second temps, le plan de développement de ce projet : découpage du projet en tâches et planning associé.

Ce projet, proposé par l'équipe Ordonnancement et Conduite (OC) au Laboratoire Information de l'école d'ingénieur universitaire polytechnique Tours est encadré par Faiza Sadi et Ameur Soukhal qui représente la MOA. Le projet sera réalisé par Baptiste Mille, élève ingénieur DI5 Polytech Tours qui représente la MOE.

1. *PFE : Projet de Fin d'Études*

1.2 Contexte de la réalisation

1.2.1 Contexte

Un problème d'ordonnancement classique consiste à organiser un nombre de tâches, en respectant des contraintes temporelles et de ressources dans le but d'optimiser une fonction objectif.
Exemple : n travaux à ordonnancer sur 2 machines, le critère à minimiser est le maximum des dates de fin d'exécution des travaux : $P2||C_{max}$

Un problème d'ordonnancement multicritère diffère d'un monocritère en optimisant plusieurs fonctions objectifs au lieu d'une seule. La qualité de la solution est donc mesurée par plusieurs critères.
Exemple : n travaux à ordonnancer sur 2 machines. Le premier critère à minimiser est le maximum des dates de fin d'exécution des travaux. Le second est le minimiser la somme des travaux en retard [2] :$P2||C_{max}, \sum U_j$

Nous nous intéressons dans ce projet à une classe particulière des problèmes d'ordonnancement multi-critère. Ces problèmes sont appelés dans la littérature, ordonnancement de travaux interférants. Dans leur formulation, plusieurs agents sont considérés, tel que chaque agent possède un sous-ensemble de travaux et une fonction objectif. Un critère global est aussi considéré (agent global) afin de mesurer la qualité de l'ordonnancement appliqué sur la totalité des travaux. Celui-ci est fixé suivant la notation à trois champs des problèmes d'ordonnancement introduit par [1]. On note ces problèmes par $\alpha|\beta|f^{10}, ..., f^{1k}$ tel que :

- α est représentatif de l'environnement
- β les contraintes
- f^{10} critère global
- f^{1k} critère de l'agent 1

Si uniquement deux agents sont considérés, alors on notre f^A l'agent global et f^B l'autre agent. Quand il s'agit d'environnement à machines parallèles, ces problèmes s'avèrent être NP-difficile.

1.2.2 Objectifs

Dans cette étude nous considérerons plusieurs machines identiques et deux agents A,B. L'agent global A veut minimiser son C_{max} quant à l'agent B, il cherche à minimiser le nombre de travaux en retard $\sum U_j$ (ainsi que le problème inverse : À veut minimiser $\sum U_j$ et B veut minimiser C_{max}). Avec U_j, une fonction booléenne qui prend 1 si le travail est en retard, 0 sinon.

2. *Un travail est dit en retard si la date de fin de la réalisation de celui-ci dépasse la date de fin souhaitée.*

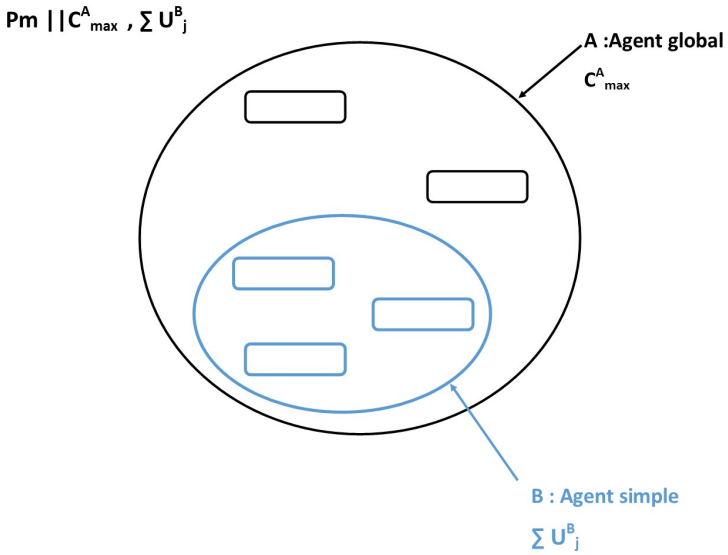

FIGURE 1.1 – Problème d'ordonnancement avec travaux interférants

Selon la formulation à trois champs $\alpha|\beta|\gamma$ des problèmes d'ordonnancements, les problèmes sujets de ce projet se notent :

$$Pm|d_i|C^{\grave{A}}_{max}, \sum U^B_i.$$
$$Pm|d_i|\sum U^A_i, C^B_{max}.$$

Étant donnée la complexité du problème $Pm||C_{max}$ et de $Pm||\sum U_j$ (problèmes NP-difficiles), la minimisation des deux critères à la fois est donc aussi NP-difficile. Une méthode exacte serait coûteuse en temps machine. Dans ce travail nous allons favoriser les méthodes heuristiques. Ces méthodes ont un meilleur rapport qualité/temps de calcul que les méthodes exactes, même si elles ne garantissent pas l'optimalité, néanmoins elles fournissent rapidement une solution dite "approchée".

On est dans un cas multicritère. Nous allons chercher l'ensemble des solutions non dominées ou un ensemble représentatif de celui-ci : front de Pareto. Ce front sera comparé à celui retourné par une méthode exacte. Puisque nous nous intéresserons à l'énumération de front de Pareto nos deux problèmes peuvent s'écrire de cette manière :

$$Pm|d_i|P(C^A_{max}, \sum U^B_i)$$
$$Pm|d_i|P(\sum U^A_i, C^B_{max})$$

Dans un premier temps, nous considérerons l'approche ϵ-contrainte. Par cette approche, nous chercherons une solution non-dominée dite solution optimale de Pareto. Il s'agit de chercher une solution minimisant le critère de l'agent A tout en bornant supérieurement la valeur de la fonction objectif de l'agent B. Les problèmes se notent donc :

$$Pm|d_i, \sum U^B_i \leq Q|C^{\grave{A}}_{max}$$
$$Pm|d_i, C^B_{max} \leq Q|\sum U^A_i$$

Ces problèmes d'ordonnancement retournent une solution optimale de Pareto. Afin d'obtenir un front de Pareto il faudra faire varier la valeur Q en suivant l'algorithme ci-dessous :

Algorithm 1 ϵ-approch

```
Q = UB
while Q ≥ LB do
    Résoudre Pm|d_i, ∑U_i^B ≤ Q|C_max^A
    if Pas de solution then
        break ;
    else
        noter(α, β) = (∑U_j^B, C_max^A) la solution retournée.
    end if
    Poser Q = α − 1
    S = SU(α, β)
end while
Retourner "S"
```

1.3 Description générale

1.3.1 Environnement du projet

Ce projet ne dépend d'aucun environnement logiciel et matériel. Il n'a donc nul besoin d'être intégré dans un environnement spécifique.

1.3.2 Caractéristiques des utilisateurs

La MOA sera l'utilisatrice des logiciels. Celle-ci possède de bonnes connaissances en informatique et en ordonnancement.

1.3.3 Fonctionnalités et structure générale du système

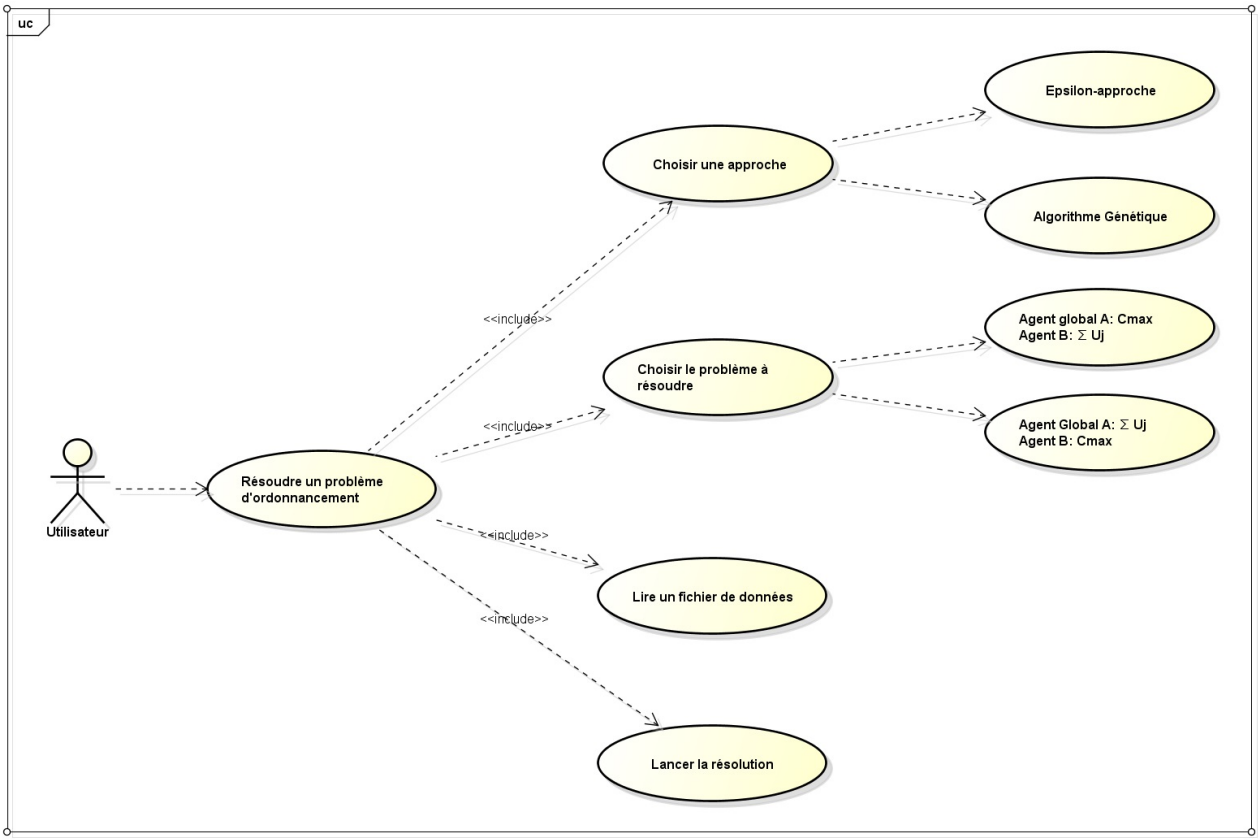

FIGURE 1.2 – UseCase résolution

Pour résoudre un problème d'ordonnancement, l'utilisateur devra renseigner l'approche qu'il souhaite utiliser (ϵ-approche ou génétique). Il choisira ensuite le problème à résoudre ($Pm|d_i|P(C^A_{max}, \sum U^B_i)$ ou $Pm|d_i|P(\sum U^A_i, C^B_{max})$). Après le traitement, les résultats seront accessibles sous forme d'un fichier texte. Ce fichier contiendra les coordonnées x et y des solutions non-dominées.

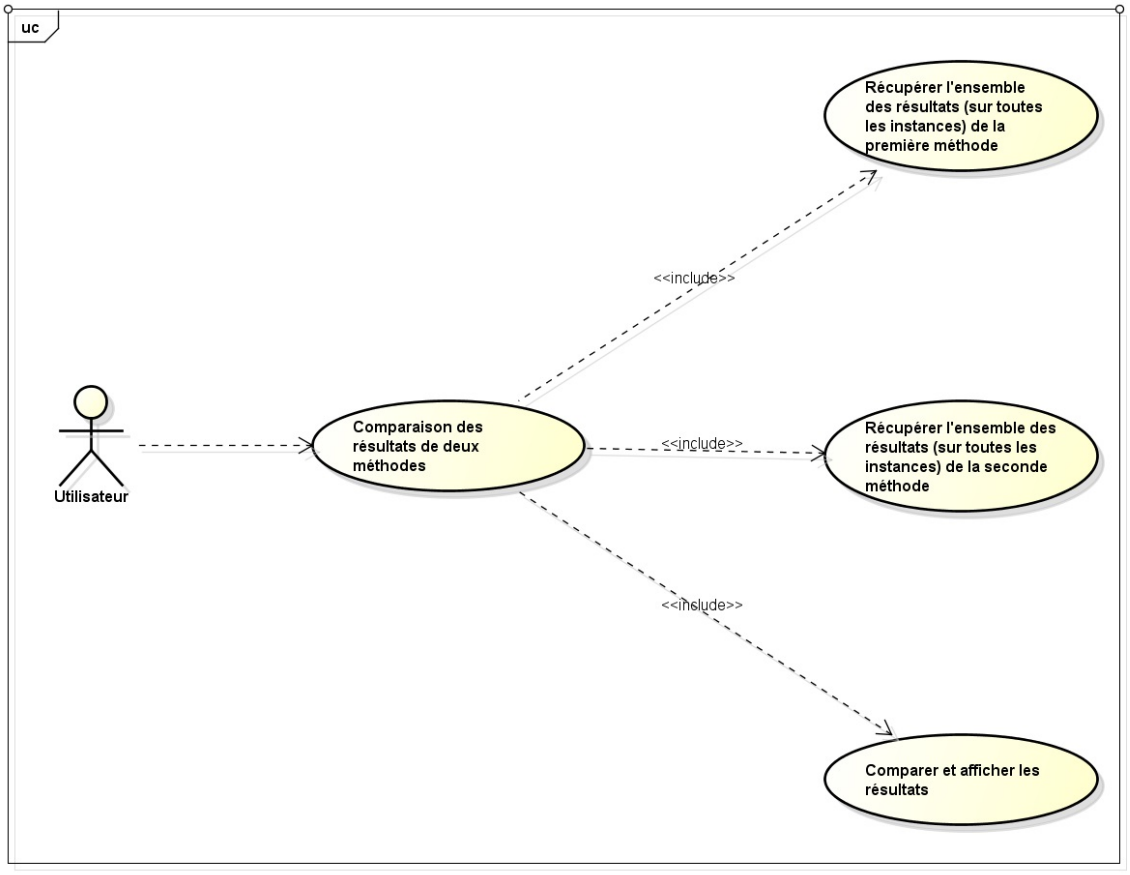

FIGURE 1.3 – UseCase comparaison de fronts de Pareto

Pour vérifier l'efficacité des heuristiques, l'utilisateur utilisera un programme de comparaison de fronts de Pareto. Après le traitement, les résultats sont accessibles sous forme de tableau dans un fichier .xls.

Afin de pouvoir récupérer facilement l'ensemble des données de chaque méthode, une convention de nommage sera à respecter. Les résultats de chaque instance devront être rangés de cette manière : chaque problème contiendra des sous-dossiers spécifiant le nombre de machines, qui contiendront des sous-dossiers indiquant le nombre de jobs qui eux-mêmes, contiendront les résultats de chaque instance et un fichier supplémentaire contenant les temps d'exécution CPU. À noter que tous les sous-dossiers devront être présents même si aucun résultat n'a pu être obtenu dans ce sous-dossier.

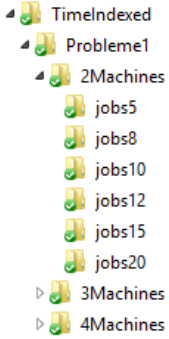

FIGURE 1.4 – Convention de nommage

1.3.4 Contraintes de développement, d'exploitation et de maintenance

Contraintes de développement

Il n'y a pas de contrainte de développement liée au matériel. Il faut noter cependant qu'une machine effectuant plus de calculs à la seconde qu'une autre résoudra le problème d'ordonnancement de manière plus rapide. Ainsi avec une machine plus "puissante", on pourra appliquer nos heuristiques sur des instances plus grandes.

Pour le programme de résolution, le langage de programmation adopté est le Java et l'IDE choisi est Eclipse. Les résultats devront être écrits dans un fichier .txt.
Pour le programme de comparaison de fronts de Pareto, celui-ci sera fait en Java avec l'IDE Eclipse. Les résultats devront être écrits dans un fichier .xls.

L'ensemble des livrables (application(s) et rapports) devront être rendus le 04/05/2014 au plus tard, et la date de soutenance sera choisie parmi : 7, 13, 14, et 15/05/2014.

1.4 Description des interfaces externes du logiciel

1.4.1 Interfaces homme/machine

Dans ce PFE, aucune interface homme/machine n'est souhaitée. Notre programme de résolution se lancera donc en ligne de commande.

1.4.2 Interfaces logiciel/logiciel

Les deux programmes à développer lors de ce PFE s'utiliseront de manière indépendante. Il n'y aura donc pas d'interaction de notre programme avec d'autres.

1.5 Architecture générale des systèmes

1.5.1 Programme de résolution

Notre programme de résolution sera lancé en ligne de commande. Voici les différentes classes qu'il contiendra :

- une classe job contenant toutes les informations relatives à un job
- une classe machine contenant toutes les informations relatives à une machine
- une classe agent contenant toutes les informations relatives à un agent
- une classe eApprochCmaxU contenant toutes les informations et méthodes relatives à l'ϵ-approche répondant au problème $Pm|d_i|P(C^A_{max}, \sum U^B_i)$
- une classe eApprochUCmax contenant toutes les informations et méthodes relatives à l'ϵ-approche répondant au problème $Pm|d_i|P(\sum U^A_i, C^B_{max})$
- une classe geneticApprochCmaxU contenant toutes les informations et méthodes relatives à l'approche génétique $Pm|d_i|P(C^A_{max}, \sum U^B_i)$
- une classe geneticApprochUCmax contenant toutes les informations et méthodes relatives à l'approche génétique $Pm|d_i|P(\sum U^A_i, C^B_{max})$
- une classe ResolveProblem permettant de gérer l'ensemble du programme.

1.5.2 Programme de comparaison de front de Pareto

Notre programme de comparaison sera lancé en ligne de commande. Voici les différentes classes qu'il contiendra :

- une classe ParetoFront contenant toutes les informations relatives à un front de Pareto.
- une classe CompareParetoFront permettant de gérer l'ensemble du programme.

1.6 Description des fonctionnalités du programme de résolution

1.6.1 Définition de la fonction 1 : dataRecovery

Identification de la fonction 1

Cette fonction permet de récupérer les données du fichier passé en entrée du programme.

Description de la fonction 1

Cette fonction prend en entrée le chemin du fichier et retourne un tableau contenant l'ensemble des données du fichier.

1.6.2 Définition de la fonction 2 : jobsCreation

Identification de la fonction 2

Cette fonction permet de créer et d'initialiser l'ensemble des jobs. Ces jobs sont des données fournies en entrée, il est donc important de les stocker afin de pouvoir récupérer leurs informations lorsque cela sera nécessaire.

Description de la fonction 2

Cette fonction prend en entrée le tableau retourné par dataRecovery et donnera en sortie une liste d'objets job. Le programme commence par créer une liste de jobs. Ensuite on ajoutera dans cette liste tous les jobs (avec leurs informations) qui sont dans le tableau. Pour finir la fonction retourne cette liste de jobs.

1.6.3 Définition de la fonction 3 : machinesCreation

Identification de la fonction 3

Cette fonction permet de créer et d'initialiser l'ensemble des machines. Ces machines sont des données fournies en entrée, il est donc important de les stocker afin de pouvoir récupérer leurs informations lorsque cela sera nécessaire.

Description de la fonction 3

Cette fonction prend en entrée le tableau retourné par dataRecovery et donne en sortie une liste d'objets machine. Le programme commence par créer une liste de machines. Ensuite on ajoutera dans cette liste toutes les machines (avec leurs informations) qui sont dans le tableau. Pour finir la fonction retourne cette liste de machines.

1.6.4 Définition de la fonction 4 : executionAlgorithm

Identification de la fonction 4

Cette fonction est présente pour chaque heuristique, elle permet de résoudre le problème avec l'heuristique en question. Les problèmes à résoudre sont les suivants :

- $Pm||P(C^A_{max}, \sum U^B_i)$
- $Pm||P(\sum U^A_i, C^B_{max})$

Description de la fonction 4

Cette fonction prend en entrée toutes les variables nécessaires à l'exécution de l'algorithme. Elle retourne soit un ensemble de solutions non dominées, dit le front de Pareto, soit une seule solution.

1.6.5 Définition de la fonction 5 : writeSolution

Identification de la fonction 5

Cette fonction écrit dans un fichier .txt le résultat obtenu par rapport à notre problème.

Description de la fonction 5

Cette fonction prend en entrée un ensemble de solutions non dominées, dit le front de Pareto et retourne un booléen en sortie indiquant si l'écriture c'est bien passée ou non.

1.7 Description des fonctionnalités du programme de comparaison de front de Pareto

Durant cette section, l'ensemble des solutions du premier front de Pareto sera noté : $FP1 = \{s_1, ..., s_e\}$ ($s_i = \{x_1, x_2\}$), tandis que l'ensemble des solutions du second front de Pareto sera noté $FP2 = \{s'_1, ..., s'_h\}$ ($s'_i = \{x'_1, x'_2\}$)

1.7.1 Définition de la fonction 1 : PercentExactSolutionFind()

Identification de la fonction 1

Cette fonction permet de calculer le pourcentage de solutions de la méthode approchée, égales à celle de la solution exacte par rapport au nombre total de solutions de la méthode approchée.

Description de la fonction 1

Cette fonction prend en entrée la sortie de la fonction 1 et la liste des solutions non dominées de la méthode approchées. Elle donne en sortie le pourcentage souhaité.
La formule mathématique correspondante est la suivante :

$$PESF = 100 * \frac{1}{h} \sum_{i=1}^{h} \sum_{j=1}^{e} \mathbb{1}_{\{s_i = s'_i\}}$$

1.7.2 Définition de la fonction 2 : AverageShorterDistance

Identification de la fonction 2

Cette fonction permet de calculer la moyenne des distances euclidiennes les plus courtes entre les solutions du premier et du second front de Pareto.

Description de la fonction 2

Cette fonction prend en entrée la liste des solutions non dominées de chaque front et donne en sortie la moyenne souhaitée.
La formule mathématique correspondante est la suivante :

$$ASD^{PE}(PH) = \frac{1}{h}(\sum_{i=1}^{h} d_i)^{1/2} \text{ avec } d_i = \sqrt{\sum_{j=1}^{2} (x_j - x'_j)^2}$$

1.7.3 Définition de la fonction 3 : AverageDistancePoints

Identification de la fonction 3

Cette fonction permet de calculer la moyenne des distances entre chacune des solutions d'un front, deux à deux.

Description de la fonction 3

Cette fonction prend en entrée la liste des solutions non dominées d'une méthode et retourne la moyenne souhaitée.
La formule mathématique correspondante est la suivante :

$$ADP^{PE}(PH) = \frac{1}{h}(\sum_{i=1}^{h} g_i)^{1/2} \text{ avec } g_i = \min_{j \in \{1...h\}} ||s'_i, s_j||$$

1.7.4 Définition de la fonction 4 : OrthogonalProjectionDistance

Identification de la fonction 4

Cette fonction permet de calculer la moyenne des plus petites distances des projections orthogonales de chaque solution du premier front de Pareto avec les segments reliant les solutions du second front de Pareto .

Description de la fonction 4

Cette fonction prend en entrée la liste des solutions non dominées de chaque front et retourne la moyenne souhaitée. La formule mathématique correspondante est la suivante :

$ADP^{PE}(PH) = \frac{1}{h}(\sum_{i=1}^{h} t_i^2)^{1/2}$ avec t_i la projection orthogonale d'une solution de FP1 sur FP2

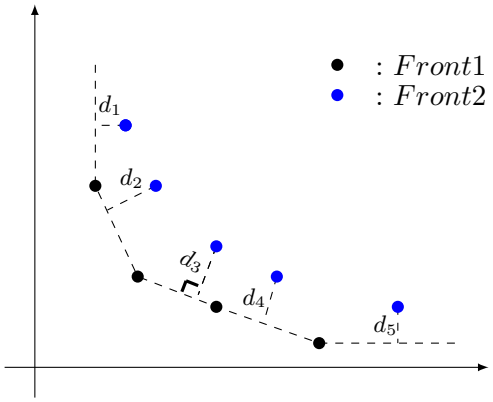

FIGURE 1.5 – Représentation de t_i : distance orthogonale entre une solution du front de Pareto de la méthode approchée et le front de Pareto exacte

1.7.5 Définition de la fonction 5 : DominantFunction

Identification de la fonction 5

Cette fonction permet de savoir si une des méthodes domine totalement l'autre ou non.

Description de la fonction 5

Cette fonction prend en entrée la liste des solutions non dominées de chaque front et retourne si une fonction domine l'autre ou non.

1.8 Conditions de fonctionnement

1.8.1 Performances

Une méthode heuristique a pour but de résoudre de manière approchée un problème plus rapidement qu'une méthode exacte. Nos heuristiques devront donc pouvoir résoudre efficacement les instances des problèmes considérés de grand taille.

1.8.2 Capacités

Les limites ne sont pas fixées, de plus elles dépendront de l'efficacité de nos algorithmes. Toutefois, il serait intéressant de pouvoir appliquer nos algorithmes sur un grand nombre d'instances. En effet, plus nous pourrons résoudre de manière efficace (front de Pareto de l'heuristique relativement proche de celui d'une méthode exacte) plus nous pourrons dire que nos algorithmes sont efficaces (i.e. "bonnes").

1.8.3 Contrôlabilité

Afin de suivre l'exécution du programme, des messages seront affichés en console.

1.8.4 Sécurité

Aucune demande particulière n'a été faite de la part de la MOA en ce qui concerne la sécurité.

1.8.5 Facteurs de Qualité

Dans une optique de futures améliorations/reprises de ce PFE, une qualité de travail est exigée. En effet, il est important de rendre des rapports bien rédigés, de la documentation et de bien commenter le code du programme. De plus ce code devra être optimisé.

Plan de développement

2.1 Découpage du projet en tâches

2.1.1 Tâche 1 : gestion et versionning du projet

Description de la tâche

Cette tâche consiste à déployer une solution pour la gestion et le versionning du projet. Pour cela un redmine ainsi qu'un svn sont mis à disposition par l'école. Si besoin, une formation sera donnée de la MOE à la MOA sur l'utilisation de ces deux outils.

Estimation de charge

Cette tâche est estimée à 0.5 jour/homme. Le temps consacré par la suite sur la mise à jour du wiki et du diagramme de Gantt n'est pas inclus dans cette estimation.

2.1.2 Tâche 2 : documentation, lecture et compréhension

Description de la tâche

Cette tâche permet de comprendre le projet et d'en définir son périmètre. Tout au long du projet, cette tâche pourra être reprise si besoin est (informations supplémentaires, par exemple). Elle est composée de lecture d'articles et de documents concernant notre sujet, mais aussi d'informations fournies par la MOA et de réunions entre la MOA et la MOE.

Livrables

Un rapport résumant l'article [2] sera à fournir, les algorithmes présents dans celui-ci serviront de base de réflexion pour nos heuristiques.

Estimation de charge

Cette tâche est estimée à 5 jours/homme - sans tenir compte d'éventuelles reprises de celle-ci -.

2.1.3 Tâche 3 : rédaction du cahier des spécifications

Description de la tâche

Cette tâche consiste en la rédaction du cahier de spécifications. Celle-ci permet de définir le contexte, le périmètre, les systèmes et leurs fonctionnalités ainsi que le déroulement du projet à l'aide d'un planning.

Livrables

Un cahier de spécification sera à rendre.

Estimation de charge

Cette tâche est estimée à 4.5 jours/homme.

2.1.4 Tâche 4 : échange du cahier des spécifications entre la MOA et la MOE

Description de la tâche

Une fois le premier jet du cahier de spécification fait, celui-ci va subir des navettes entre la MOA et la MOE afin que ces deux partis soient en accord.

Livrables

Un cahier de spécification final sera à rendre avant le 06/01/2014. À cette livraison, la MOA et la MOE auront signé ce cahier.

Estimation de charge

Cette tâche est estimée à 1 jour/homme.

Contraintes temporelles

Le livrable devra être rendu au plus tard le 06/01/2014.

2.1.5 Tâche 5 : Programme de comparaison de fronts de Pareto

Description de la tâche

Cette tâche consiste à effectuer un programme. Le but de notre programme est de pouvoir comparer deux fronts de Pareto et de retourner les résultats dans un fichier .xls.

Livrables

Un guide d'utilisateur sera à fournir.

Estimation de charge

Cette tâche est estimée à 5 jours/homme. Cette estimation prend en compte la compréhension du programme existant, le développement du nouveau programme ainsi que la rédaction du guide d'utilisateur.

2.1.6 Tâche 6 : recherche d'une heuristique basée sur l'ϵ-approche

Description de la tâche

Notre première heuristique sera basée sur l'ϵ-contrainte. Cette tâche est composée de recherche dans l'état de l'art et de création d'une heuristique appropriée à notre problème.

Livrables

Le livrable de cette tâche sera un algorithme.

Estimation de charge

Cette tâche est estimée à 6 jours/homme. Cette estimation tient compte des échanges qui seront effectués entre la MOA et la MOE sur cet algorithme.

2.1.7 Tâche 7 : Préparation soutenance mi-parcours

Description de la tâche

Préparation de la soutenance de mi-parcours

Livrables

Soutenance avec PowerPoint.

Estimation de charge

Cette tâche est estimée à 0.5 jour/homme.

2.1.8 Tâche 8 : implémentation du premier algorithme

Description de la tâche

Une fois que notre première heuristique sera validée par la MOA, il faudra l'implémenter.

Livrables

Un guide d'utilisateur sera à fournir.

Estimation de charge

Cette tâche est estimée à 3.5 jours/homme.

2.1.9 Tâche 9 : tests et correction de la première implémentation

Description de la tâche

Cette tâche consiste à effectuer de nombreux tests sur notre implémentation afin de vérifier le bon fonctionnement de l'algorithme

Livrables

Un guide d'utilisateur sera à fournir.

Estimation de charge

Cette tâche est estimée à 6 jours/homme.

2.1.10 Tâche 10 : application d'une approche génétique

Description de la tâche

Notre seconde heuristique sera basée sur une approche génétique. Cette tâche est composée de recherche dans l'état de l'art et de création d'une heuristique appropriée à notre problème.

Livrables

Le livrable de cette tâche sera un algorithme.

Estimation de charge

Cette tâche est estimée à 6 jours/homme. Cette estimation tient compte des échanges qui seront effectués entre la MOA et la MOE sur cet algorithme.

2.1.11 Tâche 11 : implémentation du second algorithme

Description de la tâche

Une fois que notre deuxième heuristique sera validée par la MOA, il faudra l'implémenter.

Livrables

Un guide d'utilisateur sera à fournir.

Estimation de charge

Cette tâche est estimée à 4 jours/homme.

2.1.12 Tâche 12 : tests et correction de la seconde implémentation

Description de la tâche

Cette tâche consiste à effectuer de nombreux tests sur notre implémentation afin de vérifier le bon fonctionnement de l'algorithme.

Livrables

Un guide d'utilisateur sera à fournir.

Estimation de charge

Cette tâche est estimée à 6 jours/homme.

2.1.13 Tâche 13 : rédaction du rapport final

Description de la tâche

Cette tâche consiste à la rédaction du rapport final de ce PFE, afin de rendre compte du projet. Celui-ci pourra être réalisé tout au long du projet.

Livrables

Le livrable de cette tâche est le rapport en lui-même.

Estimation de charge

Cette tâche est estimée à 4 jours/hommes.

Contraintes temporelles

Le livrable devra être rendu au plus tard le 04/05/2014.

2.1.14 Tâche 14 : préparation de la soutenance

Description de la tâche

Cette tâche consiste à préparer la soutenance de ce PFE avec la réalisation d'un diaporama sur le projet et les résultats obtenus, ainsi qu'une éventuelle manipulation.

Livrables

Le livrable de cette tâche est le code source, les exécutables du projet et le PowerPoint de la soutenance.

Estimation de charge

Cette tâche est estimée à 2 jours/hommes.

Contraintes temporelles

La soutenance se déroulera lors d'une de ces dates : 7, 13, 14, et 15/05/2014.

2.2 Planning

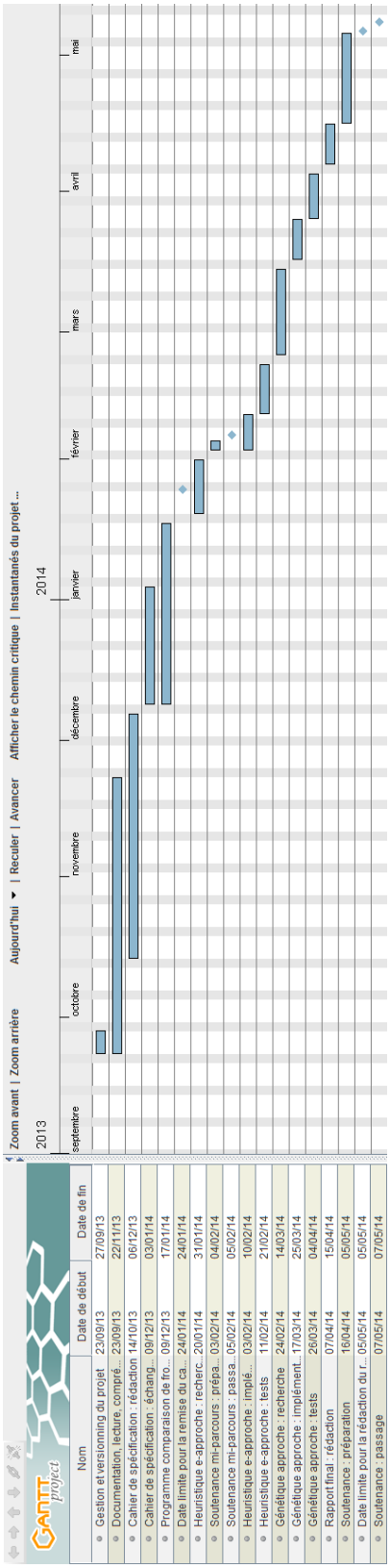

FIGURE 2.6 – Diagramme de Gantt

Bibliographie

[1] Jean-Charles Billaut and Vincent T'kindt. Multicriteria scheduling. *Springer*, 2006.

[2] Johnny C.Ho and Yih-Long Chang. Minimizing the number of tardy jobs for m parallel machines. *ELSEVIER*, 1995.