KEVIN BROWN

UN ALGORITHME HYBRIDE D'OPTIMISATION EN GESTION MULTI-PROJETS

MÉMOIRE PRÉSENTÉ

À L'UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

POUR L'OBTENTION

DU GRADE DE MAÎTRISE ÈS SCIENCES (M. SC.)

DÉPARTEMENT DES SCIENCES ADMINISTRATIVES

UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

PROGRAMME DE MAÎTRISE EN GESTION DE PROJET

FÉVRIER 2014

Résumé

La gestion multi-projets gagne de plus en plus de popularité grâce à sa flexibilité et son efficience par rapport à l'utilisation des ressources. En effet, ces dernières étant les principales contraintes des projets, un programme permettant de les rassembler donnerait l'occasion aux entreprises de mieux les partager. La concurrence agressive et la situation économique actuelle forcent l'optimisation de l'usage des ressources. L'accélération de projets et l'allocation des ressources sont maintenant des atouts essentiels aux gestionnaires de programmes afin d'assurer la survie des projets.

Cette recherche a pour but d'outiller le gestionnaire de programmes en ce qui a trait à l'optimisation de multi-projets. Plus spécifiquement, l'objectif de cette recherche est de proposer un algorithme hybride permettant d'accélérer les projets. Cet algorithme hybride est formé par l'amalgamation de l'algorithme tabou et du modèle mathématique de programmation linéaire de Tissier (2008). L'algorithme tabou est un méta heuristique qui aidera le modèle mathématique à accélérer les projets. L'algorithme hybride tire son efficience de sa rapidité d'exécution, de sa flexibilité par rapport au programme de projets ainsi que de sa capacité. Il permet une accélération grâce à la réallocation des ressources, qui provient de la décélération des activités non critiques. Cette décélération permet une économie en capital qui sera alors réinvestie dans l'accélération des activités critiques.

Une étude de cas a été faite afin de valider l'algorithme hybride. Elle réutilise les même trois projets que Tissier (2008) avait utilisés, afin de permettre la comparaison des résultats. L'algorithme a permis d'accélérer le programme à l'objectif voulu tout en obtenant une diminution des coûts de 6,92 % par rapport au modèle de Tissier (2008).

Signatures	
de l'Étudiant :	du Directeur :

Table des matières

Résumé 2
Liste des tables 5
Table des figures 6
Introduction 7
Chapitre 1 : Cadre conceptuel 11
1.1 Concepts 11
1.2 Classe de problème NP-Difficile (NP-Hard) 17
1.3 Problématique 18
Chapitre 2 : Revue de littérature 20
2.1 La gestion multi-projets 20
2.2 Problème d'allocation des ressources et d'accélération de projets 22
2.3 Modèle de décomposition Dantzig-Wolfe 31
2.4 Solution hybride 32
Chapitre 3 : Modélisation mathématique 33
3.1 Définition 34
3.2 Modélisation mathématique 36
3.3 Algorithme tabou 43
3.4 L'approche hybride 46
Chapitre 4 : Application de l'algorithme sur une étude de cas 61
4.1 Présentation du problème 62
4.2 Présentation des résultats 74
4.3 Comparaison des résultats avec Tissier (2008) 80
CONCLUSION 84

Index 87
Bibliographie 88
Annexe A.....91
Annexe B.....106

Liste des tables

Table 3.1 Table des variables.....40
Table 3.2 Table des paramètres.....41
Table 4.1 Activités du projet A.....65
Table 4.2 Caractéristiques financières du projet A.....66
Table 4.3 Activités du projet B.....68
Table 4.4 Caractéristiques financières du projet B.....69
Table 4.5 Activités du projet C.....71
Table 4.6 Caractéristiques financières du projet C.....72
Table 4.7 Finance du programme.....73
Table 4.8 Sommaire financier du programme du programme original.....76
Table 4.9 Sommaire des résultats.....77
Table 4.10 Possibilité restante pour accélérer et décélérer les activités.....79
Table 4.11 Résultat de la solution finale.....80
Table 4.12 Sommaire financier du programme de la solution finale.....80
Table 4.13 Comparaison des résultats avec Tissier (2008).....82
Table B1 Chemin critique de la solution initiale.....110
Table B2 Activité accélérée et décéléré de la solution initiale.....112
Table B3 Sommaire financier du programme lors de la solution initiale.....113
Table B4 Activité accélérée pour l'itération deux.....115
Table B5 Activité accélérée pour l'itération dix.....117

Table des figures

Figure 3.1 Schématisation de l'algorithme hybride.....49
Figure 3.2 Solution initiale.....52
Figure 3.3 Trouver la meilleure activité.....53
Figure 3.4 Sélection de la solution.....55
Figure 4.1 Diagramme PERT du projet A.....67
Figure 4.2 Diagramme de PERT du projet B.....70
Figure 4.3 Diagramme de PERT du projet C.....73
Figure 4.4 Diagramme PERT du programme.....75

Introduction

L'unicité des projets dicte les besoins spécifiques ainsi que les contraintes particulières de ceux-ci. Les ressources sont une des contraintes majeures, et une certaine compétition existe pour la distribution de celles-ci. Les projets manquant de ressources ne verront jamais le jour tandis que ceux qui ont de meilleurs directeurs et outils auront une meilleure chance d'obtenir des ressources et ainsi d'assurer leur survie. En gestion de projet, les compagnies s'inspirent des projets réussis afin de recréer le même succès.

Dans une entreprise, tous les projets sont importants. Certains projets ayant un lien en commun sont rassemblés sous le même « grand projet », qui est appelé programme, d'où l'origine de la gestion des programmes, qui permet de partager les ressources au lieu de procéder à une compétition. La gestion de programme est donc une adaptation visant à assurer la survie d'une plus grande partie des projets dans une entreprise. Cette évolution n'est pas la gestion de projet appliqué à plusieurs projets; elle demande l'adoption d'une toute nouvelle approche. Le portefolio, quant à lui, est le management (la gestion) de plusieurs projets n'ayant aucun lien en commun, mais dirigés par le même directeur. Il permet donc de redistribuer les ressources entre les différents projets.

Ce qui est semblable entre la gestion de programme et la gestion de portefolio est que pour atteindre les buts organisationnels, l'allocation des ressources est cruciale (Görög 2011). Le terme multi-projets rassemble les deux catégories.

Lors de l'étape de la planification d'un projet, l'allocation des ressources est souvent source de conflits. Dans un environnement multi-projets, les ressources, qui prennent la forme de capital humain, de budget et de temps, seront précieusement distribuées entre les différents projets. Puisque ces ressources sont très limitées, certains projets privilégiés auront la plus grosse part du gâteau et d'autres seront mis sur une liste d'attente. Plusieurs outils applicables à la gestion de projet comme le diagramme de GANTT et l'analyse PERT sont utilisés pour des projets simples. Mais est-ce que ces outils sont encore adaptés au style de gestion? Ils ne présentent pas d'algorithme ayant la capacité d'analyser les projets en parallèle afin de répartir les ressources tout en s'adaptant à la situation.

En effet, le temps est une ressource sur laquelle il n'est pas toujours évident de mettre un prix. Le triangle temps-budget-qualité est une notion enseignée dans les rudiments de la gestion de projet. Afin de d'obtenir des contrats où la concurrence est élevée, une accélération de projet permet d'optimiser l'horaire pour éviter des délais inutiles. En réduisant la durée du projet et du programme, la compagnie peut obtenir une

prime importante. Dans un programme ou un portefolio, plusieurs projets en parallèle se chevauchent et créent une opportunité d'effectuer une optimisation. Afin d'optimiser un projet, les tâches du chemin critique seront visées pour réduire la durée totale de celui-ci. Lorsqu'on accélère un projet, il y a habituellement un coût associé, car la consommation de ressources est plus élevée. Par contre, lorsqu'un projet est en attente en raison de circonstances externes et non contrôlables, il peut y avoir des ressources humaines qui ne sont pas utilisées à leur maximum. Ceci étant dit, on peut avoir un ajustement et une redistribution constante des ressources afin d'accélérer certains projets à moindre coût. Cette flexibilité vient en partie du cadre de gestion multi-projets. D'autres techniques peuvent aussi être utilisées, comme la décélération et la sous-traitance de projet.

Tissier (2008), qui s'est inspiré du travail de Wiley, Deckro et al. (1998), propose deux modèles mathématiques : le premier a pour but d'accélérer un programme de projet (à moindres coûts). Le second modèle permet l'allocation des ressources humaines selon les résultats du premier modèle. Cette recherche a pour but d'optimiser les programmes de projets en appliquant un algorithme tabou au modèle mathématique, créant ainsi un algorithme hybride. Tissier (2008) ainsi que Wiley, Deckro et al. (1998) ont gardé le modèle le plus général possible afin de l'appliquer à une multitude de projets. Aussi, une étude de cas est réalisée dans le but de démontrer la validité de l'algorithme avec des variables réelle. L'étude de cas sera réalisée dans un

environnement non spécifique puisque l'algorithme hybride sera développé de façon générale. Les projets n'auront pas de facteurs de priorité et seront traités comme égaux. En gardant l'algorithme hybride général, il sera utile à plus de gestionnaires de programmes.

Cette recherche porte sur la gestion multi-projets et traite spécifiquement de l'accélération de projets ainsi que de l'allocation des ressources. L'objectif général de cette recherche est de développer un outil pour aider les gestionnaires de multi-projets à optimiser leurs projets sur les plans d'accélération et d'allocation de ressources. Plus spécifiquement, la présente recherche applique un algorithme tabou au modèle mathématique de Tissier (2008) afin d'augmenter l'efficience du modèle et de diminuer son temps d'exécution.

Le premier chapitre de ce document présente le cadre conceptuel de la recherche, c'est-à-dire la problématique ainsi que les définitions nécessaires à la compréhension du texte. En deuxième partie, une revue de littérature est effectuée sur les concepts impliqués dans cette recherche. Le troisième chapitre comporte le développement de l'algorithme hybride en intégrant l'algorithme tabou au modèle mathématique servant à l'accélération des projets. Le quatrième chapitre présente les résultats de l'étude de cas ainsi que la validation du modèle. Finalement, la conclusion résume le travail de recherche et présente des opportunités pour les prochains développements.

Chapitre 1 : Cadre conceptuel

Dans ce chapitre, nous expliquons les concepts importants pour cette recherche et présentons la problématique en détail. Bien que cette recherche soit effectuée dans un programme de gestion de projets, elle regroupe certains concepts de plusieurs autres domaines tels que la recherche opérationnelle et les mathématiques. Les concepts présentés dans ce cadre conceptuel sont la gestion multi projet, l'optimisation et le type de problème NP-Hard.

1.1 Concepts

Multi-projets

Avec la récente crise économique et la diminution de la quantité de ressources naturelles comme le pétrole, la gestion de projet prend de plus en plus d'ampleur. En effet, ces facteurs dictent l'élimination des gaspillages de ressources, d'où l'importance d'une gestion efficace. Puisqu'il y a des temps morts dans un projet, les gestionnaires ont décidé de regrouper plusieurs projets sous un même groupe afin de partager les ressources. Ce partage de ressources permet un meilleur contrôle et, par le fait même,

optimise l'efficacité. Voyons plus spécifiquement ce que certains termes signifient lorsqu'on parle de rassemblement de projets : multi-projets, portefolio et programme.

Programme : Un programme de projet est semblable au portefolio. Par contre, les projets sont reliés par un but commun.

Portefolio : Un portefolio constitue un ensemble de ressources géré par un groupe qui est distribué selon les besoins des projets. Les projets peuvent avoir un objectif commun, mais ceci n'est pas un préalable. Un portefolio est donc une technique que les gestionnaires ont mise en place afin d'améliorer la gestion des ressources ainsi que l'efficience de la compagnie.

Projet simple : Généralement de très grosse taille, les projets simples sont de plus en plus rares.

Multi-projets : La gestion multi-projets signifie par définition qu'un gestionnaire de projets coordonne plusieurs projets. De nos jours, la gestion de projets simples ne constitue qu'un faible pourcentage de la réalité.

Aux fins de la présente recherche, nous utiliserons le terme multi-projets. Aussi, les accords de libre-échange entre pays augmentent le nombre de compagnies en compétition. Au gouvernement, un processus de « bid » est obligatoire afin de sélectionner une compagnie lors d'un achat ou d'un projet. Les compagnies doivent donc soumissionner et la plus performante gagnera. Les compagnies ne peuvent se permettre de soumissionner à un prix sans profit ou marge de manœuvre, d'où l'importance d'optimiser le projet afin de battre le marché.

Optimisation

Afin d'être parmi les plus compétitives, les compagnies doivent maximiser l'utilisation de leurs ressources, sans pour autant les utiliser à l'excès. Mais où peut-on optimiser un projet? Le triangle vertueux nous apporte cette réponse : le temps, le budget et la qualité. Ce triangle, qui est né avec la gestion de projet, démontre les relations entre les trois composantes cruciales. Certes, nous ne voulons pas optimiser un projet en coupant dans la qualité, ceci serait une contradiction. Il ne reste que deux options : le temps et le budget. Afin d'optimiser le temps et le budget, deux solutions nous paraissent évidentes : l'accélération de projet ainsi que l'allocation de ressources.

Accélération/décélération/sous-traitance

L'accélération de projets ou de tâches signifie de raccourcir le temps d'exécution de celui-ci, ce qui engendre généralement un coût. Cependant, le fait de terminer un projet avant la date prévue peut permettre d'obtenir des bonus de la part du client, qui peuvent être utilisés pour couvrir les frais d'accélération. Aussi, chaque jour, les faux frais reliés au projet s'accumulent et sont également réduits lorsqu'un projet est accéléré. Afin d'accélérer un projet, il faut accélérer les tâches sur le chemin critique.

Par exemple, si une personne produit un soulier par jour, et qu'il faut produire dix souliers au total, la tâche prendra dix jours. En ajoutant une autre personne, il y aura deux fois plus de souliers seront produits; la tâche prendra donc cinq jours. Toutefois, il se peut que la deuxième personne ait besoin d'une courbe d'apprentissage et ne soit pas aussi efficiente au début. De plus, certaines tâches ne peuvent être accélérées en raison de leur nature.

D'autres tâches, qui ne sont pas sur le chemin critique du projet, peuvent être décélérées afin de diminuer certains coûts. Par exemple, si je dispose de dix jours pour confectionner dix souliers et que mon équipe compte deux personnes, je peux enlever une personne de cette équipe et soit la réassigner à une autre équipe, soit mettre fin à son

contrat. D'autres tâches peuvent être sous-traitées à des compagnies dans le but de travailler en parallèle.

Allocation de ressources

Dans un projet, il y a trois types de ressources : humaines, budgétaires et matérielles. Le temps n'est pas considéré comme une ressource, car nous ne sommes pas maîtres de l'horaire des livrables. Le gestionnaire doit donc manipuler les trois autres ressources afin de respecter l'horaire et de minimiser les délais. Dans les projets de technologie de l'information, les ressources humaines sont les plus importantes. En effet, il est beaucoup plus difficile de remplacer un programmeur qu'un ordinateur.

L'allocation de ressources, ou la réallocation de ressources, est un facteur clé de la réussite. Chaque fois que l'on accélère ou décélère une tâche, une réallocation de ressources va avoir lieu.

La majorité des projets utilisent l'unité de mesure « man-hour » dit heure-personne afin d'estimer une quantité de travail. Par exemple, si une personne peut produire un soulier en huit heures, la quantité de travail associée à la production d'un

soulier est de huit heures-personnes. Mathématiquement, on peut donc déduire que le temps de réalisation d'une tâche et le nombre d'employés font partie d'une équation linéaire inversement proportionnelle : plus on augmente le nombre d'employés, moins le temps de réalisation est élevé. Malheureusement, ce n'est pas aussi simple; il y a des restrictions.

Toutefois, certaines ressources sont uniques et ne peuvent être interchangées, ce sont les spécialistes. Un des avantages de la gestion multi-projets est qu'un spécialiste peut travailler sur plusieurs projets en même temps et maximiser son utilisation du temps. En effet, il arrive qu'un projet soit en attente ou que la tâche effectuée en ce moment ne requière pas l'usage du spécialiste. Celui-ci est alors payé à ne rien faire s'il travaille sur un seul projet. Il est à noter qu'un spécialiste est généralement payé cher.

Lorsqu'on joue avec les ressources humaines, il faut faire très attention, car il ne s'agit pas seulement de budget. On ne peut pas licencier du personnel pour un mois et ensuite le réengager, d'où le but de maximiser l'utilisation de l'équipe. Le but de cette recherche n'est pas de discuter de la prise de décision ou du côté éthique.

1.2 Classe de problème NP-Difficile (NP-Hard)

Le titre NP-Hard définit une classe de problèmes très précise souvent appliquée à l'optimisation. L'abréviation NP signifie « nondeterministic polynomial time » et représente une catégorie de problèmes décisionnels qui peut être résolue par une machine Turing. « Nondeterministic » dit que l'algorithme solution sera de forme essai et erreur. Polynomial time est le nom donné à un algorithme pour lequel le temps d'exécution est limité par une expression polynomiale de la grosseur des variables d'entrée pour l'algorithme. L'algorithme solution prend donc un temps d'exécution qui est exponentielle au nombre d'entrées. « Hard » est l'adjectif utilisé pour désigner que le problème NP est aussi difficile que le plus difficile des problèmes NP. Donc un algorithme NP-Hard se doit de résoudre le plus difficile des problèmes NP. À ce jour, nous n'avons toujours pas découvert de solution exacte pour cette classe de problème vu la complexité.

Un exemple typique de problème NP-Hard est celui du « travelling salesman ». Un vendeur faisant du porte-à-porte doit se rendre dans un certain nombre de villes pour effectuer ses ventes. On se demande ensuite quel trajet sera le plus rapide. Bien qu'il soit facile de trouver la solution optimale à ce problème s'il n'y a que dix villes, nous n'avons pas la solution optimale pour X villes. Dans le cas de la gestion de projet, on peut simplifier la situation en remplaçant le vendeur par l'équipe de projet et les villes

par les tâches à effectuer. Les types de problèmes NP-Hard sont à la base de plusieurs autres problèmes allant de la gestion des avions à l'intelligence artificielle. Le problème d'accélération de projet et d'allocation de ressource est considéré NP-Hard.

1.3 Problématique

Dans le champ de la gestion de projet, les directeurs de projets se retrouvent souvent sans outil efficient pour optimiser leur projet. Certes, plusieurs logiciels comme Microsoft Projet existent sur le marché, mais leurs algorithmes d'optimisation sont considérés comme archaïques. En ce moment, les gestionnaires de projets compensent ce manque grâce à leur expérience et font des choix sans avoir eu la chance d'explorer toutes les possibilités.

L'objectif général de cette recherche est de développer un outil pour aider les gestionnaires de multi-projets à optimiser leurs projets sur les plans d'accélération et d'allocation de ressources. L'impact sera observé à la fois au niveau financier et au niveau de la réputation de l'entreprise. Cette recherche a pour but d'appliquer un algorithme tabou au modèle mathématique développé par Wiley, Deckro et al. (1998) et

amélioré par Tissier (2008) afin de créer un algorithme hybride ayant pour but d'augmenter l'efficience du modèle et diminuer son temps d'exécution.

Plus spécifiquement, l'algorithme hybride devra :

1. accélérer le programme de projets;
2. diminuer le temps d'exécution, c'est-à-dire résoudre le problème en faisant moins d'itération;
3. minimiser les coûts reliés au programme de projets.

Une étude de cas sera réalisée afin de tester le nouveau modèle mathématique. Il sera ensuite comparé au résultat du modèle de Tissier (2008).

Chapitre 2 : Revue de littérature

La croissante compétition entre les entreprises, entraînée par le libre-échange est maintenant au niveau mondial. Cette compétition amène la gestion de projet à gagner plus d'ampleur au sein des compagnies. Ceci étant dit, afin de gagner un contrat, les compagnies doivent avoir un plan moins coûteux et plus rapide d'exécution que leur compétiteur. Ce chapitre propose une revue de littérature sur la gestion multi-projets.

2.1 La gestion multi-projets

Le concept de multi-projets a été développé afin de faciliter la gestion de plusieurs projets relevant d'une même équipe. Selon Payne (1995), l'enseignement de la gestion de projets simples joue un rôle purement académique puisque ce type de gestion ne représente pas la majorité des cas. En effet, il est utile d'apprendre les rudiments de la gestion de projet avant de se lancer dans un programme multi-projets, où la complexité est exponentielle. De plus, 90 % des projets se situent dans un environnement multi-projets, ce qui signifie qu'une simple amélioration dans ce domaine amènerait des bénéfices énormes (Payne 1995).

Aussi, avec la situation économique mondiale, les entreprises ne peuvent se permettre d'avoir des ressources qui sont sous-utilisées. Les projets sont habituellement indépendants, mais les ressources proviennent d'un bassin qui appartient au programme. Les plus gros projets et les plus critiques reçoivent en effet les meilleures ressources alors que les plus petits ont de la difficulté à survivre. Il est cependant difficile d'utiliser les ressources humaines à 100 % tout le temps, ce qui amène un nouveau problème : la surexploitation des ressources. Puisque les programmes multi-projets contiennent des projets qui sont généralement plus courts et moins importants, les ressources humaines tendent à faire des « rushs » leur demandant de travailler sur plusieurs projets en même temps, ce qui entraîne leur surutilisation (Payne 1995). Ces problèmes d'allocation des ressources sont fréquents et sont des enjeux clés en gestion de projet.

La formalisation des projets et des portefolios est souvent l'objet de débats; pas assez de formalisation diminue significativement le pouvoir du gestionnaire et trop de formalisation encombre inutilement le système d'administration. Teller, Unger et al. (2012) étudient l'impact que la formalisation des projets et du portefolio ont sur le succès de ce dernier. Puisque la formalisation accroît la transparence du portefolio, ce qui ensuite facilite l'allocation des ressources, les conflits reliés aux ressources sont diminués (Teller, Unger et al. 2012). De plus, Teller, Unger et al. (2012) démontrent que la formalisation est fortement corrélée avec la vitesse d'allocation des ressources et la fiabilité de l'engagement. Ils contribuent au champ de la gestion multi-projets en

renseignant les gestionnaires sur l'importance de la formalisation afin augmenter la performance et le succès de leur portefolio.

Jonas (2010) présente un article sur le pouvoir des gestionnaires de portefolios, qui est un poste récent dans le leadership des compagnies. Paradoxalement, le gestionnaire de portefolios détient des responsabilités énormes, mais ne dispose que d'une faible autorité; il doit gérer les ressources communes et les attribuer aux projets qui compétitionnent entre eux. Jonas (2010) catégorise l'implication des gestionnaires de portefolios (en attribuant du pouvoir, en intervenant et en encourageant), il définit leurs spécifications de travail et schématise leurs tâches. Dans son modèle, il considère l'aspect temporel comme critères de succès de gestion de portefolio. Jonas (2010) conclut que le gestionnaire de portefolios est positivement corrélé au succès du portefolio en plus d'avoir un impact direct sur la qualité des tâches et un impact indirect sur la performance. Jonas (2010) modélise bien le rôle du gestionnaire de portefolios, cependant, il ne développe pas d'outil afin de les aider dans leurs tâches.

2.2 Problème d'allocation des ressources et d'accélération de projets

Le problème de la planification des projets (project scheduling problem [PSP]) est apparu avec la popularisation de la gestion de projets. En effet, planifier l'horaire de

chaque tâche est le travail ardu du gestionnaire de projets. Plusieurs chercheurs se sont attardés à ce problème, qui a évolué dans le temps. Selon Payne (1995), l'accès à des outils efficaces pour planifier des programmes de multi-projets est très difficile. Les algorithmes sont en effets compliqués et difficiles à utiliser. Nous ne disposons pas d'outils efficaces pour planifier les projets et les outils d'optimisation sont plutôt rares.

Engwall et Jerbrant (2003) ont effectué une recherche dans le but de cerner les problèmes opérationnels communs à la gestion multi-projets ainsi que les causes de ces derniers. Ils procèdent à une étude de cas sur deux entreprises différentes dans le domaine de la technologie de l'information. Engwall et Jerbrant (2003) découvrent que le « syndrome d'allocation des ressources » est en effet commun à la gestion multi-projets qui est créé par la dépendance de plusieurs projets à une seule ressource clé. Les projets du programme entrent alors en compétition pour accaparer la ressource rare. Un autre facteur de ce syndrome est le manque de ressources supplémentaires, qui est entraîné par une trop grande quantité de projets. Conséquemment, lorsqu'un projet va mal, tous les autres sont affectés, car les solutions à court terme mises en œuvre dictent souvent d'augmenter les ressources du projet problème aux dépens des autres. Le syndrome d'allocation des ressources serait dû à une mauvaise planification et un manque de ressources. Engwall et Jerbrant (2003) démontrent bien l'importance de la planification des ressources, mais leur étude ne propose pas d'outils pour résoudre le problème.

Fréquemment, les gestionnaires accélèrent les projets afin de gagner un contrat. Yaghootkar et Gil (2012) étudient, à l'aide d'une simulation, l'impact de la réallocation des ressources lorsqu'elles sont toutes utilisées dans le cadre d'un projet assujetti à un horaire strict. Selon Yaghootkar et Gil (2012), les projets devant respecter un horaire strict (schedule-driven) mettent de la pression sur les ressources. Leurs résultats prouvent que les gestionnaires de multi-projets devraient toujours garder en réserve des ressources spécialisées. Lorsqu'un projet prend du retard et qu'il faut lui réallouer des ressources, cela retarde les autres projets et l'effet peut être catastrophique pour une compagnie. La recherche de Yaghootkar et Gil (2012) démontre bien l'importance d'avoir des ressources spécialisées libres, mais elle ne propose pas d'outils pour aider les gestionnaires à bâtir un système.

RCPSP (Resource constrain project scheduling project)

En 1950, le diagramme de PERT fait son apparition. Ce diagramme sert à illustrer les différentes tâches d'un projet, en démontrant un lien de préséance entre chacune d'entre elles pour enfin trouver la durée du projet. En effet, on peut ensuite trouver le chemin critique, et par le fait même, les tâches critiques. Cette méthode, qui ne prend en considération que le temps (la durée) de chaque tâche, fonctionne très bien dans un monde idéal. Cependant, dans la réalité, ce n'est pas le cas. Certaines

ressources, souvent humaines, sont irremplaçables soit en raison de leur expérience, soit de leur rareté. Ces ressources restreintes provoquent des délais. Ce problème particulier est ce qu'on appelle le « resource-constrained project scheduling problem » (RCPSP) (Montoya-Torres, Gutierrez-Franco et al. 2010).

Ce problème est caractérisé comme « NP-hard » (non-deterministic polynomial-time hard). Ce problème d'optimisation NP-hard demande de trouver le chemin le moins coûteux et ne peut être résolu par des calculs mathématiques à cause de la complexité du problème. Ce problème est souvent représenté par le problème du vendeur qui doit choisir le chemin optimal entre multiples destinations. Afin de résoudre ce problème, les chercheurs ont développé des méthodes heuristiques, c'est-à-dire non exactes.

Plusieurs chercheurs se sont intéressés au problème RCPSP en gestion de projet. Bien que ce problème traite d'allocation de ressources et d'optimisation de coûts, il traite seulement de projets et non de multi-projets. Certains principes sont à conserver, mais la différence est trop grande pour être appliquée à la gestion de multi-projets.

Multi-mode resource-constrained project scheduling problem (MRCPSP)

Le problème de gestion de projets mutifonctions à contraintes de ressources (multi-mode resource-constrained project scheduling problem [MRCPSP]) est similaire au précédent. Coelho et Vanhoucke (2011) font une distinction claire entre les algorithmes qui comprennent des algorithmes pour les ressources renouvelables et non renouvelables ainsi que les ressources non renouvelables.

Les lacunes du projet simple

Ces problèmes font partie de la gestion de projet simple. Bien que certains concepts de la gestion multi-projets soient adaptables à la gestion de projet, l'inverse n'est pas vrai. L'allocation des ressources et l'accélération en gestion de projet simple comportent plusieurs problèmes (Browning et Yassine 2010). La simplicité de ces derniers enlève le réalisme des algorithmes développés et l'interdépendance des projets amène des difficultés supplémentaires. Heureusement, quelques chercheurs ont développé des algorithmes sur les programmes multi-projets.

Resource constrained multi project scheduling problem (RCMPSP)

Ce problème consiste à prioriser les activités afin d'optimiser une fonction qui réduit les délais des projets ainsi que du programme (Browning et Yassine 2010). Comme dans les problèmes précédents, le RCMPSP est considéré NP-hard, alors les chercheurs ont développé des méthodes heuristiques puisqu'il n'y a pas d'algorithme connu pour le résoudre. De plus, ce problème s'inscrit dans un concept de multi-projets, ou les recherches ont eu moins d'attention lors des dernières années et où les gestionnaires n'ont pas beaucoup d'outils (Browning et Yassine 2010).

Selon Browning et Yassine (2010), les solutions heuristiques comportent quatre grandes catégories de solution : les règles de priorités, les métas heuristiques classiques (algorithme génétique et tabou), les non-standards métas heuristiques et les métas heuristiques mélangés.

Une des méthodes heuristiques les plus utilisées dans la littérature est les règles de priorité (PR). Browning et Yassine (2010) ont choisi de tester toutes les règles de priorité et de déterminer lesquelles sont les plus efficaces dans certains types de projet. Lorsqu'un problème NP-Hard est particulièrement gros, la méthode des règles de priorités est préférée étant donné la simplicité de la solution et la facilité de mise en

œuvre (Browning et Yassine 2010). Une autre raison de l'utilisation des règles de priorité ainsi que de sa popularité est que les gestionnaires de projets ne construisent pas toujours les réseaux PERT, qui sont essentiels aux autres méthodes heuristiques. Alors, face à des décisions reliées à l'allocation de ressources, les gestionnaires de projets vont utiliser un mélange d'intuition et de règle générale (Browning et Yassine 2010). Toutefois, cette méthode n'est qu'un amalgame d'expérience de gestionnaires de projet qui est traduit en règle et ne correspond pas à une optimisation. Krüger et Scholl (2009) trouvent la solution du problème avec des règles de priorité de sélection des tâches ainsi que des règles de transfert de ressources.

Une seconde approche est les méthodes méta heuristiques classiques. Les algorithmes génétiques ainsi que l'algorithme de recherche tabou font partie de cette catégorie. L'algorithme tabou génère une solution aléatoire de départ pour ensuite analyser les solutions voisines et décider si elles sont plus performantes (Nkubili, Kane et al. 2011). Gendreau (2002) a écrit une bonne introduction résumant les principes de base et avancés pour l'algorithme tabou. Gonçalves, Mendes et al. (2008) ainsi que Montoya-Torres, Gutierrez-Franco et al. (2010) ont travaillé sur des algorithmes génétiques. Les algorithmes génétiques fonctionnent en générant une population de solutions, sélectionnant les meilleures pour ensuite les reproduire et finalement créer une nouvelle population de solutions, qui à son tour, recommencera le même processus

(Gonçalves, Mendes et al. 2008). L'algorithme tabou semble être le plus populaire pour résoudre les problèmes d'optimisation.

La troisième catégorie de solution inclut les algorithmes d'agent, ils sont non standard métaheuristiques. Confessore, Giordani et al. (2006) ont développé un système multi agent. Chaque agent possède de l'information différente sur l'environnement. Grâce à cette information, il peut ensuite élaborer une stratégie et compétitionner avec les autres pour l'allocation des ressources (Confessore, Giordani et al. 2006). Ce système fonctionne par itération et la solution est trouvée lorsque tous les agents gagnent.

La quatrième catégorie se nomme « solution méta heuristique diverse ». Elle se compose de solutions « forward backward ». Rabbani, Fatemi Ghomi et al. (2007) ont développé un modèle (Backward pass method) dont la tâche est de minimiser la multiplication de la durée et la variance du projet en déterminant le temps de fin de chaque activité au lieu du temps de commencement. (Rabbani, Fatemi Ghomi et al. 2007).

Optimisation multi objectifs (allocation de ressources)

Plusieurs autres problèmes tournent autour du même sujet d'optimisation et d'allocation des ressources. Le problème d'optimisation multi objectif (et d'allocation de ressources) représente bien la réalité et il n'y a pas de solution optimale qui existe (Rabbani, Fatemi Ghomi et al. 2007). Cependant, Azaron, Katagiri et al. (2006) se sont concentrés sur le « trade off » coût/durée afin de développer un algorithme qui permet l'allocation des ressources dans le diagramme PERT.

Bien que ces problèmes traitent de la planification du temps et des ressources, ils visent à minimiser les délais lorsqu'il y a un manque de ressources. Les problèmes précédents sont très importants pour la gestion de projet, car les délais font souvent la norme. Cependant, ma recherche nuance ces problèmes, car elle traite de la planification et bien que je tente d'optimiser le coût et la durée des projets, je considère que les ressources ne sont pas illimitées ou en pénurie. De plus, mon but est de développer un modèle mathématique afin d'obtenir une solution qui pourra être utile aux gestionnaires de projet. Bien que les méthodes heuristiques soient très bien pour trouver de bonnes solutions pour les gros problèmes NP-Hard, elles ne peuvent trouver de solution exacte. L'algorithme de décomposition Dantzig-Wolfe sera utilisé à la base et ensuite modifié par une heuristique.

2.3 Modèle de décomposition Dantzig-Wolfe

L'algorithme Dantzig-Wolfe a été développé spécialement afin de trouver une solution parfaite à un problème d'optimisation à plusieurs variables et contraintes (Dantzig et Wolfe 1960). Wiley, Deckro et al. (1998) ont choisi cet algorithme afin d'accélérer un programme de projet tout en optimisant les coûts. Wiley, Deckro et al. (1998) se sont aperçus que dans un programme, certaines contraintes s'appliquaient au programme de projet (budget du programme, ressources humaines et limite de temps) alors que d'autres contraintes s'appliquaient seulement aux projets à l'intérieur du programme (préséance des projets, limites d'accélération, budgets du projet et ressources humaines du projet). Les contraintes du programme forment le problème maître alors que les contraintes des projets forment les sous-problèmes. À partir du WBS (work breakdown structure), l'algorithme de Dantzig-Wolfe permet d'obtenir la solution optimale (Wiley, Deckro et al. 1998). Cette solution exacte est possible puisque le problème est limité dans sa taille. En augmentant la taille du problème, les calculs deviennent trop lourds et le modèle peut être adapté en solution heuristique. Il faut donc relaxer le problème. L'avantage de ce modèle est qu'il permet d'obtenir une solution exacte optimale à petite échelle et une solution adéquate à grande échelle. Tissier (2008) a repris le travail de Wiley, Deckro et al. (1998) afin de l'actualiser et de l'améliorer. Premièrement, Tissier (2008) actualise le modèle de Wiley, Deckro et al. (1998) en rendant possible la sous-traitance des projets et en ajoutant les contraintes d'accélération et de décélération. Deuxièmement, Tissier (2008) amène un second modèle traitant de

l'allocation des ressources humaines dans le cadre d'un programme de multi-projets. Toutefois, il reste de la place à l'amélioration. Les deux modèles de Tissier (2008) peuvent être adaptés pour accepter un grand nombre de tâches.

2.4 Solution hybride

Comme nous l'avons lors de cette revue de littérature, deux types de solutions sont produites; exactes et heuristiques. Les méthodes exactes sont trop strictes et ne permettent pas de répondre au besoin des gestionnaires de projets puisqu'un programme peut facilement comprendre de 120 à 480 tâches (Krüger et Scholl 2009). Récemment, les nouveaux algorithmes dits hybrides utilisent un mélange des deux solutions. Le principe est de garder une solution exacte, tout en prenant certains raccourcis. Une approche est de créer une heuristique afin de résoudre un sous-problème dans la stratégie de recherche pour une méthode exacte (Jourdan, Basseur et al. 2009). Dans plusieurs disciplines, comme la recherche opérationnelle, les solutions hybrides démontrent de bons résultats, mais en gestion multi-projets, cette approche n'est presque pas exploitée. Vasquez (2001) a développé un puissant algorithme hybride impliquant de la programmation linéaire et un algorithme tabou afin de résoudre le problème 0-1 multi knapsack, qui comporte beaucoup de similitudes avec l'optimisation de projet. Cette recherche a pour but de développer un algorithme hybride impliquant de la programmation linéaire (Dantzig-Wolfe) et un algorithme tabou afin de résoudre le problème d'accélération et d'allocation des ressources en multi-projets.

Chapitre 3 : Modélisation mathématique

Ce chapitre présente la modalisation mathématique du problème ainsi que l'agencement avec l'algorithme tabou. L'algorithme hybride accélère les programmes de multi-projets tout en gardant les coûts dans les limites des contraintes. Il accélère les tâches critiques et décélèrent les tâches non critiques afin d'optimiser globalement le programme. Chaque projet, à l'intérieur du programme, à ses propres contraintes de temps, budget et de jour d'accélération maximum, qui sont respectées par l'algorithme.

Le but de cet algorithme est d'aider les gestionnaires de multi-projets à planifier et à contrôler leur projet, ainsi qu'à devenir plus efficients dans la gestion de ce dernier. Le chapitre est divisé en trois parties. La première partie montre le modèle mathématique utilisé pour développer l'algorithme. La deuxième partie explique l'utilisation de l'algorithme tabou. La troisième partie montre la coopération entre le modèle mathématique et l'algorithme tabou ainsi que l'intégration de ces derniers.

3.1 Définitions

Les définitions suivantes s'ajoutent à celles du chapitre un. Elles sont très utiles afin de garder une certaine uniformité et de faciliter la lecture.

Algorithme tabou : L'algorithme tabou est un méta heuristique servant à rechercher la meilleure solution possible dans un ensemble de solutions trop grand qui rend impossible l'exploration de toutes les options.

Modélisation mathématique : La modélisation mathématique est la représentation mathématique de la réalité. Dans cette recherche, elle représente un programme de multi-projets avec toutes ses contraintes.

Algorithme hybride : L'algorithme hybride constitue une amalgamation de l'algorithme tabou au modèle mathématique.

Itération : Une itération est un cycle de recherche complet par l'algorithme hybride. À chaque itération, l'algorithme hybride obtient une solution différente, qu'elle soit validée ou non, retenue ou rejetée.

Activité : Une activité représente une tâche dans un des trois projets. Une activité ou tâche peut être accélérée ou décélérée. L'algorithme tabou cherche à apporter des changements à une activité ou à une tâche.

Changement : Un changement est le remplacement d'un nœud par l'algorithme tabou. Dans ce contexte d'optimisation de projet, il peut s'agir soit d'une accélération d'une journée d'une tâche critique, soit d'une décélération d'une journée d'une tâche non critique.

Solution : Une solution est une réponse trouvée par l'algorithme hybride, elle peut être valide ou non-valide et optimale ou non-optimale.

Voisinage : Un voisinage représente l'ensemble de solutions qui peuvent être obtenues en effectuant un seul changement.

Optimum local : Un optimum local est la meilleure solution dans un certain voisinage. L'algorithme ne pourra donc plus s'améliorer en n'effectuant qu'un seul changement.

3.2 Modélisation mathématique

Le modèle mathématique a pour objectif de représenter la réalité tout en démontrant les équations nécessaires à l'optimisation de la durée du programme de projets. Le modèle mathématique montre aussi les limites des contraintes financières de chaque projet et du programme utile à l'accélération des activités sur le chemin critique et la décélération des activités qui ne sont pas considérées comme critiques. Ce modèle est celui de Wiley et al (1998) avec les modifications de Tissier (2008). Tissier (2008) a remis à jour le modèle mathématique de façon à représenter la réalité à laquelle font face les gestionnaires de projets. Aucune autre modification n'a été apportée au modèle mathématique. Les variables ainsi que la notation restent les mêmes afin de faciliter la continuité et de favoriser de futures améliorations.

Il est important de noter les quatre prémisses suivantes qui ont été adoptées par Wiley et al (1998) et Tissier (2008). Elles sont conservées afin d'assurer l'uniformité du modèle mathématique et ainsi avoir une comparaison valable des résultats.

Prémisse 1 : Les durées normales de réalisation des activités sont connues et prédéfinies. Ceci représente le plan initial du gestionnaire de projets, c'est-à-dire que le temps d'accomplissement des tâches n'est pas accéléré ou décéléré.

Prémisse 2 : L'accélération d'une activité est permise, par réduction de sa durée jusqu'à une certaine limite d'unité de temps. Le gestionnaire connaît donc la limite maximale de temps dans le cadre de laquelle les tâches peuvent être accélérées.

Prémisse 3 : La décélération d'une activité est permise, par augmentation de sa durée jusqu'à une certaine limite d'unité de temps. Le gestionnaire connait donc la limite maximale de temps dans le cadre de laquelle les tâches peuvent être décélérées.

Prémisse 4 : Le temps et le coût sont liés par une fonction linéaire dans le cadre de leur variation. En temps réel, le temps et le coût sont reliés par une fonction convexe. Ce modèle mathématique simplifie le calcul en faisant une approximation grâce à une fonction linéaire. Donc, en accélérant une tâche, les coûts augmentent proportionnellement et lorsque l'on décélère, les coûts diminuent de manière proportionnelle.

Variable

Trois variables sont utilisées dans le modèle mathématique. Ces trois variables ont été reprises du modèle de Wiley et al (1998) :

Table 3.1

Table des variables

x_i	Date de réalisation critique du nœud *i*
y_{ij}	Quantité maximale d'unités de temps dont l'activité (*i,j*) peut être accélérée
z_{ij}	Quantité maximale d'unités de temps dont l'activité (*i,j*) peut être décélérée

Les détails du programme et des projets doivent être insérés par le gestionnaire. Ces informations ne changeront pas. Les paramètres du modèle de Tissier (1998) ont été repris.

Table 3.2

Table des paramètres

B	Budget du programme
B_k	Budget du projet k
OH	Frais généraux du programme, par unité de temps
OH_k	Frais généraux du projet k, par unité de temps
T	Date normale d'achèvement du programme, par unité de temps
TCD	Date accélérée d'achèvement du programme (objectif), par unité de temps
T_k	Date accélérée d'achèvement du projet k (objectif), par unité de temps
R	Prime d'avance sur la date d'achèvement du programme, par unité de temps gagnée
n	Identifiant du nœud final du programme
dk	Identifiant du nœud du début du projet k
I	Ensemble des arcs (i,j) dans le programme
I_k	Ensemble des arcs (i,j) dans le projet k; I_k élément de I, pour tout k, un arc représente une activité dans la représentation des réseaux PERT
T_{ij}	Durée normale de l'activité (i,j)
D_{ij}	Coût normal de l'activité (i,j)
K_{ij}	Coût d'accélération de l'activité (i,j), par unité de temps
M_{ij}	Limite supérieure de y_{ij}, quantité maximale d'unités de temps dont on peut

	accélérer l'activité (i,j)
A_{ij}	Coût de décélération de l'activité (i,j), par unité de temps
N_{ij}	Limite supérieure de z_{ij}, quantité maximale d'unités de temps dont on peut décélérer l'activité (i,j)
P	Limite maximale du coût d'accélération global du programme
P_k	Limite maximale du coût d'accélération global du projet k

Fonction objective

La fonction objective suivante représente la minimisation à effectuer.

$$Min\ z = \sum_k OH_k\,(x_n - x_{dk}) + (x_n - T)R + \sum_i \sum_j K_{ij} y_{ij} - \sum_i \sum_j A_{ij} z_{ij}$$

Les contraintes suivantes représentent la réalité à laquelle le programme de projet fait face.

1. Contrainte pour le budget général du programme :

$$0 < \sum_k OH_k\,(x_n - x_{dk}) - (TDC - x_n)R + \sum_i \sum_j D_{ij} + \sum_i \sum_j K_{ij} y_{ij} - \sum_i \sum_j A_{ij} z_{ij}$$

$$\leq B\ ,\ avec\ (i,j) \in I$$

2. Contrainte représentant l'accélération ou la décélération au niveau du programme :

$$0 \leq \sum_i \sum_j K_{ij} y_{ij} - \sum_i \sum_j A_{ij} z_{ij} \leq P, \forall\, (i,j) \in I$$

3. Contrainte représentant la préséance entre les activités du programme

$$x_j - x_i + y_{ij} \geq T_{ij}\,, avec\ (i,j) \in I, si\ l'activité (i,j) ne\ peut\ être\ qu'accélérée,$$

$$x_j - x_i - z_{ij} \geq T_{ij}\,, avec\ (i,j) \in I, si\ l'activité (i,j) ne\ peut\ être\ que\ décélérée.$$

4. Contrainte représentant la date de fin du programme et le départ au plus tard du projet k du programme :

$$x_n \leq TCD$$

$$x_n - x_{dk} \leq T_k, \forall k\ et\ avec\ (i,j) \in I_k$$

5. Contrainte représentant le budget du projet k :

$$0 < OH_k (x_n - x_{dk}) + \sum_i \sum_j D_{ij} + \sum_i \sum_j K_{ij} y_{ij} - \sum_i \sum_j A_{ij} z_{ij} \leq B_k\,, avec\ (i,j) \in I_k$$

6. Contrainte représentant la préséance des activités du projet k :

$$x_j - x_i + y_{ij} \geq T_{ij}\,, avec\ (i,j) \in I_k\,, si\ l'activité(i,j) ne\ peut\ être\ qu'accélérée,$$

$$x_j - x_i - z_{ij} \geq T_{ij}\,, avec\ (i,j) \in I_k, si\ l'activité(i,j) ne\ peut\ être\ que\ décélérée.$$

7. Contrainte représentant l'accélération ou la décélération d'activité du projet k :

$$0 \leq \sum_i \sum_j K_{ij} y_{ij} - \sum_i \sum_j A_{ij} z_{ij} \leq P_k\,, \forall k\ et\ avec\ (i,j) \in I_k$$

8. Contrainte représentant l'accélération maximale de l'activité (i, j) :

$$y_{ij} \leq M_{ij}\,, avec\ (i,j) \in I$$

9. Contrainte représentant la décélération maximale de l'activité (i, j) :

$$z_{ij} \leq N_{ij}\,, avec\ (i,j) \in I$$

10. Contrainte représentant la positivité des variables :

$$x_i \geq 0$$

$$y_{ij} \geq 0$$

$$z_{ij} \geq 0$$

3.3 Algorithme tabou

L'algorithme tabou est une heuristique utilisée afin de générer une réponse presque optimale au problème NP-Hard. L'efficience de cet algorithme réside dans le fait qu'il n'explore pas toutes les solutions possibles, car elles sont trop nombreuses. Il se concentre donc sur les solutions qui ont un certain potentiel. L'algorithme tabou fonctionne en plusieurs étapes. L'algorithme tabou commence par trouver une solution initiale au problème qui ne demande pas d'effort, mais qui n'est toutefois pas optimale. Ensuite, l'algorithme procède à un changement par rapport à la solution initiale et vérifie si le changement améliore la réponse. Si le changement améliore la réponse, le changement est alors accepté et remplace la solution. Si le changement n'améliore pas la solution, il est considéré comme tabou et ne pourra pas être répété. C'est aussi de cette façon que l'algorithme discrimine les solutions qui ont du potentiel de celles qui n'en ont pas. Ce processus est itératif et arrête lorsqu'un des critères est atteint.

1. Représentation de la solution et du problème : La représentation du problème est faite grâce à la modélisation mathématique comprenant la fonction objective et les contraintes.

2. Solution initiale : La solution initiale est le point de départ de l'algorithme tabou. Elle est trouvée avec l'approche avare. L'approche avare trouve

d'abord la tâche critique la moins chère à accélérer afin d'accélérer le projet à moindre coût. Il procède de cette façon jusqu'à ce qu'il ait accéléré toutes les activités critiques possibles dans la limite de budget. Il décélère aussi les activités non critiques afin de réinvestir cette somme d'argent dans l'accélération des activités critiques.

3. Voisinage : Le voisinage loge toute nouvelle solution qui est atteinte en ne faisant qu'un seul changement dans la solution présente. Par exemple, une fois que la solution initiale est trouvée, l'algorithme tabou regarde les changements possibles afin d'améliorer la solution. L'ensemble de solutions dans le voisinage est créé à partir des activités pouvant être accélérées.

4. Liste tabou : La liste tabou est créée afin d'empêcher que l'algorithme tourne en rond dans un petit ensemble de solutions. Les activités accélérées brisant les contraintes ou étant contre-productives sont identifiées comme tabou et ne peuvent plus être choisies dans les futures itérations.

5. Critère d'aspiration : La liste tabou peut être trop forte et empêcher une certaine évolution qui serait critique. Parfois, il faut faire un changement contre-productif afin de pouvoir se trouver dans un nouveau voisinage avec

un nouvel ensemble de solutions disponibles, qui seront plus productives. Ce présent algorithme applique le critère d'aspiration avec la méthode suivante; si le changement est sur la liste tabou, l'algorithme est autorisé à continuer et à vérifier si le prochain changement est productif. Dans le cas où le deuxième changement est productif, les deux changements seront acceptés et l'algorithme continue son chemin normal. Dans le cas où le deuxième changement est non productif ou non acceptable, le premier changement reste sur la liste tabou et ne sera plus considéré.

6. Critère de diversification : L'algorithme tabou peut se retrouver bloqué dans un optimum local, c'est-à-dire que le processus ne trouve plus de meilleure solution dans son voisinage immédiat à cause des restrictions et contraintes imposées, même si le problème peut encore être optimisé. Le critère de diversification permet alors de changer complètement le voisinage. Lorsque l'algorithme ne peut plus trouver un changement capable d'améliorer la solution, celui-ci va resituer la solution dans un nouveau voisinage. Aussi, après un certain nombre d'itérations, si aucun changement améliorant la situation n'est trouvé, un changement aléatoire sera fait afin d'explorer une nouvelle région.

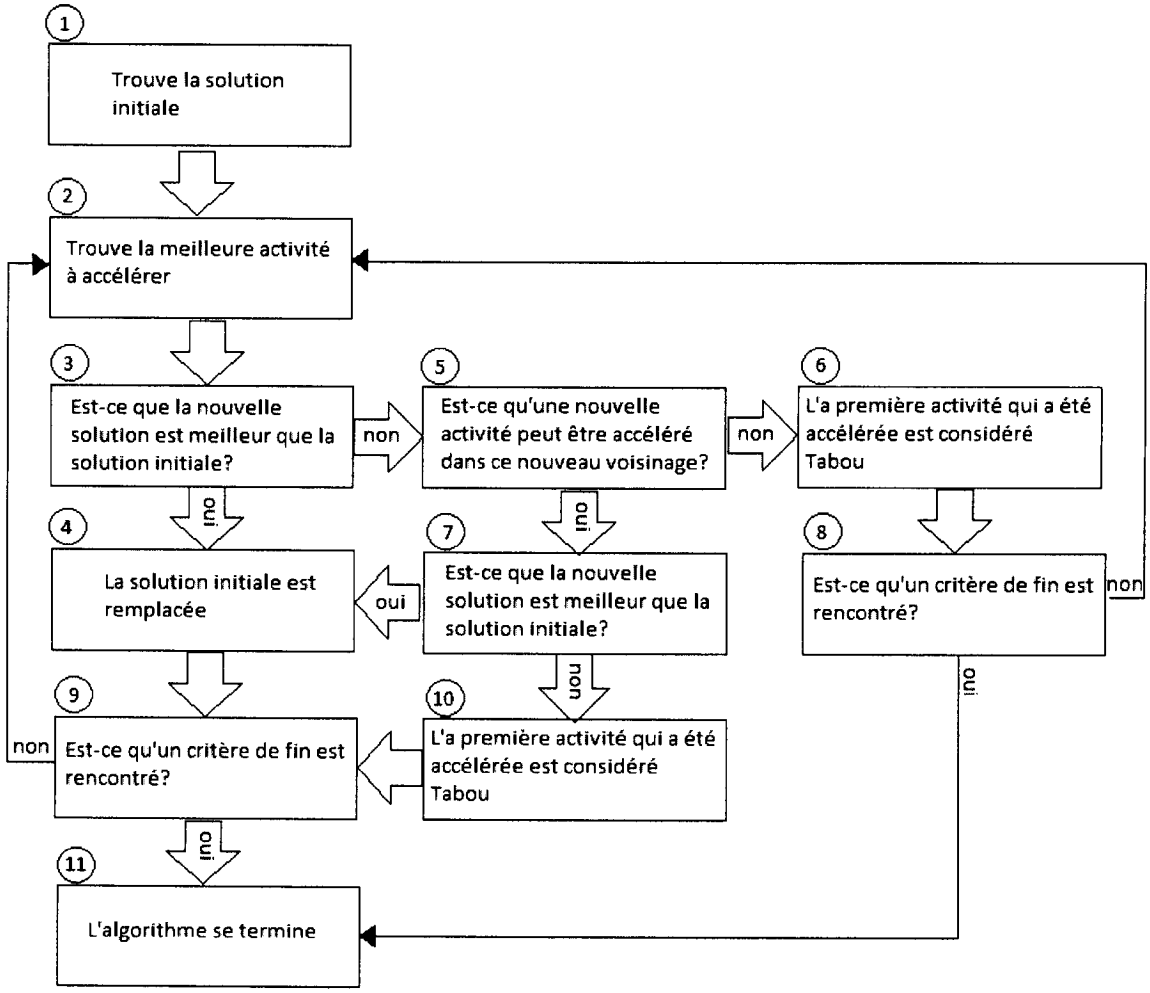
7. Critère de fin : Le critère de fin de l'algorithme est un nombre d'itérations spécifiques. Aussi, si le gestionnaire de projets spécifie une durée spécifique à atteindre, l'algorithme se terminera après avoir accéléré le programme jusqu'à cet objectif et optimisé le budget.

3.4 L'algorithme hybride

L'algorithme hybride est présenté à l'image 3.1 avec un très haut niveau d'abstraction. La fonction objective de minimisation n'apparaît pas dans l'algorithme hybride. Comme avec la décomposition Dantzig-Wolfe, ce sont les contraintes rassemblées en sous-problèmes qui génèrent des solutions.

Figure 3.1

Schématisation de l'algorithme hybride



Explication détaillée de l'intégration

Les différentes contraintes de la modélisation mathématique sont intégrées avec l'algorithme tabou afin de faire un hybride. La section suivante explique chaque partie de l'algorithme hybride en détail, tout en montrant les différentes intégrations des contraintes de la modélisation mathématique ainsi que des critères de l'algorithme tabou.

1. Trouver la solution initiale.

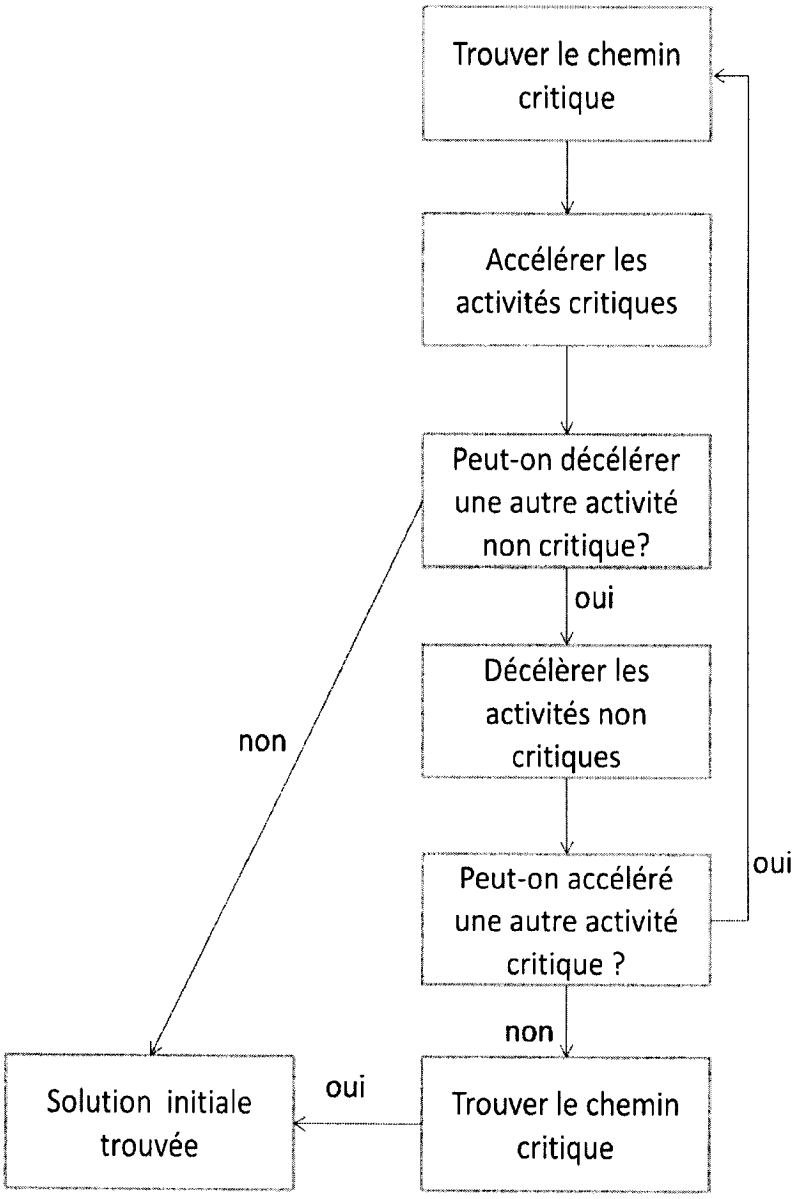
La solution initiale de l'algorithme hybride est la première étape. Elle commence par trouver le chemin critique avec les contraintes 4 et 7. La contrainte 4 permet de garantir que le programme de projet se déroule dans un ordre logique et chronologique tandis que la contrainte 7 garantit que les activités à l'intérieur de chaque projet sont exécutées dans l'ordre logique dicté par le gestionnaire de projets. La contrainte 4 tient aussi compte des activités interprojet.

Une fois que le chemin critique a été trouvé, l'algorithme accélère les activités critiques les moins coûteuses en premier et décélèrent ensuite les activités non critiques. Les contraintes 3 et 8 sont utilisées.

L'algorithme travaille en itération, c'est-à-dire que lorsque le budget est insuffisant pour accélérer la prochaine activité critique, il recherche des activités non critiques à décélérer afin d'augmenter le budget du programme et des projets. Ensuite, il continue à accélérer les activités jusqu'à ce qu'il ne puisse plus respecter les contraintes de budget. À ce moment, la solution initiale est trouvée, car le programme de projet est accéléré au maximum, mais sans l'utilisation de l'algorithme tabou.

Figure 3.2

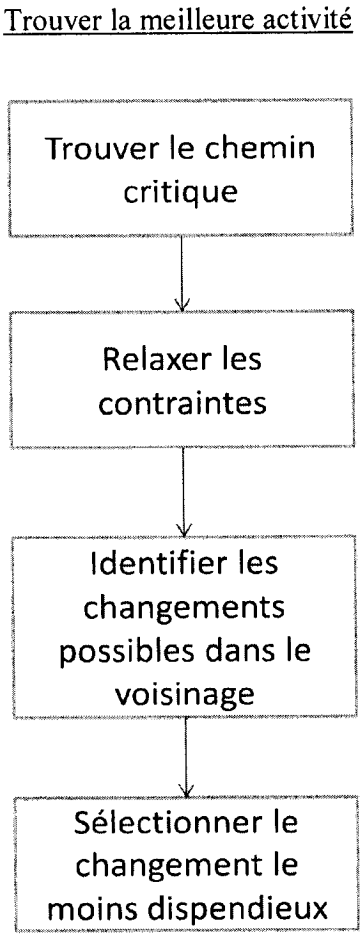
Solution initiale



2. Trouver la meilleure activité à accélérer.

Une fois que la solution initiale a été trouvée, l'algorithme cherche à s'améliorer jusqu'à ce qu'il satisfasse à un critère de terminaison. Dans l'ensemble des activités, celles qui peuvent être encore physiquement accélérées feront partie du voisinage. Pour déterminer laquelle il tentera d'accélérer en premier, l'algorithme regarde celles qui sont sur le chemin critique. Puisque la solution initiale est conçue pour qu'aucune tâche critique ne soit accélérée sans briser une des contraintes de budget, l'algorithme doit choisir celle qui semble briser le moins de contraintes pour le moment. Les contraintes sont donc relaxées à cette étape afin de permettre à l'algorithme de progresser.

Figure 3.3



3. Est-ce que la nouvelle solution est meilleure que la solution initiale?

À l'étape trois, l'algorithme compare la nouvelle solution trouvée avec l'ancienne solution (dans le cas de la première itération, il s'agit de la solution initiale). Avant de comparer la nouvelle solution avec l'ancienne, l'algorithme doit tester si la nouvelle solution est acceptable, puisqu'à l'étape précédente,

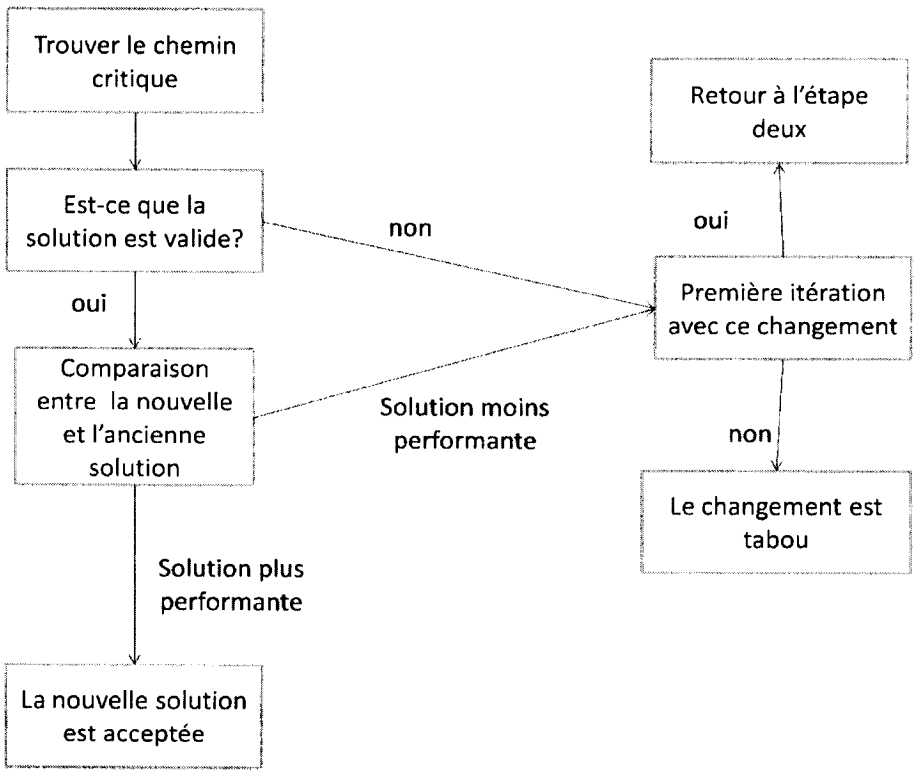
certaines contraintes ont été relaxées. La nouvelle solution doit être testée face à toutes les contraintes;

- les contraintes de budget de projet;
- les contraintes de budget de programme;
-les contraintes d'accélération/de décélération de tâche maximale;
-les contraintes d'accélération/de décélération maximale de projet; et
-les contraintes d'accélération/de décélération maximale de programme.

Une fois que la nouvelle solution est validée, la comparaison entre la nouvelle et l'ancienne peut commencer. Le facteur de comparaison est la durée du programme. À cette étape, l'algorithme trouve le chemin critique de la nouvelle solution et le compare à celui de l'ancienne solution. Si le chemin critique de la nouvelle solution est plus court, la solution est donc acceptée. Ceci est le principe fondamental de l'accélération de multi-projets. À cause de la nature de l'algorithme, il est normal que lors de la première itération, la nouvelle solution ne soit pas meilleure que la solution initiale. Ceci est dû au fait que la solution initiale comporte un certain niveau d'optimisation, car elle s'arrête lorsqu'elle ne voit plus de changement immédiat capable d'améliorer la solution. Il faudra donc au moins deux changements consécutifs avant d'obtenir une nouvelle solution meilleure que la solution initiale, d'où l'importance de relaxer les contraintes à l'étape deux.

Figure 3.4

Sélection de la solution



4. La solution est remplacée.

Cette étape est très simple. Une fois que l'algorithme à confirmer que la nouvelle solution est valide et plus performante que l'ancienne, elle est entreposée aux mêmes endroits que l'ancienne solution. L'ancienne solution est donc effacée définitivement.

5. Est-ce qu'une nouvelle activité peut être accélérée dans ce nouveau voisinage.

Après chaque changement (accélération ou décélération d'activité), le voisinage doit changer et s'adapter. Le but de cette fonction est de trouver une nouvelle activité dans un tout nouveau voisinage. Elle est nécessaire afin d'améliorer la solution initiale, car la solution initiale est optimisée dans son entourage immédiat. En effet, lors de la première itération, cette étape jouera un grand rôle afin de ne pas bloquer l'algorithme dans un voisinage immédiat. Ce changement de voisinage a été programmé pour ce faire seulement deux fois. Par contre, il reste flexible. Dans le cas d'un problème avec plus de contraintes, il sera impératif d'augmenter le nombre de changements de voisinage possibles, puisque moins d'options de changement seront disponibles. Aussi, comme à l'étape 3, cette étape relaxe les contraintes de sélection afin d'obtenir plus d'options. Les contraintes seront resserrées lors de la comparaison de la nouvelle solution avec la solution initiale.

6. La première activité qui a été accélérée est considérée comme tabou.

Premièrement, une liste est créée dès le lancement de l'algorithme, il s'agit d'un élément clé au succès de ce dernier. La liste tabou est simplement une allocation de mémoire dédiée à enregistrer les restrictions d'accélération de certaines activités qui ne peuvent améliorer la solution dans un avenir proche. L'algorithme se base alors sur la formation actuelle de la solution afin de

déterminer si l'accélération d'une activité peut améliorer la solution dans un avenir proche. Si l'algorithme détermine que le changement n'aidera pas la solution, l'accélération de cette activité sera inscrite sur la liste tabou afin de réduire le nombre de solutions possibles à explorer et aussi à réduire le voisinage. Cette étape est importante, puisque dans un problème NP-Hard, le nombre de solutions possibles augmente trop vite pour permettre la vérification de chacune d'entre elles. Ceci étant dit, le nombre de solutions est restreint et le nombre d'itérations nécessaire pour trouver la solution est plus petit. De plus, lorsqu'une activité est inscrite sur la liste tabou, elle y reste seulement pour un temps prédéfini, qui lui aussi est flexible et dépendant de la taille du problème. De cette façon, on empêche de bloquer un changement qui pourrait être utile dans le futur, avec une autre configuration de la solution (en raison de l'évolution de celle-ci.).

7. Est-ce que la nouvelle solution est meilleure que la solution initiale?

   Cette étape est la même que l'étape trois.

8. Est-ce qu'un critère de fin est respecté?

   Il est important de choisir judicieusement les critères de fin de l'algorithme. Sans ces derniers, le programme pourrait faire des itérations

jusqu'à ce que l'ordinateur ne soit physiquement plus capable de fonctionner. Les critères de fin sont les suivants :

Itération maximale : Un nombre d'itérations maximal est spécifié afin d'empêcher que le programme tourne en boucle trop longtemps. Ce critère est spécifié par le gestionnaire de projets et dépend du nombre de tâches du programme de projets. Plus il y a de tâches, plus d'itérations seront nécessaires afin de trouver une réponse satisfaisante.

Objectif atteint : Ce critère permet de spécifier un objectif à atteindre au programme. Par exemple, le gestionnaire de projets peut spécifier de combien de jours il a besoin d'accélérer le programme de projets avec un budget spécifique. Si cet objectif est atteint, le programme s'arrête et retourne la réponse.

La solution n'évolue pas : si la solution est bloquée dans un optimum local, elle peut cesser d'évoluer. Le gestionnaire de projets doit ensuite retourner modifier le nombre d'itérations qui dictent les changements de voisinage avant de considérer une activité tabou. Il est aussi possible que l'algorithme ait trouvé la meilleure solution, ou une solution extrêmement bonne qui est difficile à remplacer. Dans ce cas, le gestionnaire a sûrement spécifié un grand nombre d'itérations maximales.

9. Est-ce qu'un critère de fin est respecté?

   Cette étape est la même que l'étape huit.

10. La première activité qui a été accélérée est considérée comme tabou.

    Cette étape est la même que l'étape six.

11. L'algorithme se termine.

    Après avoir rencontré un critère de fin, l'algorithme s'assure de retourner la durée de chaque tâche, le chemin critique, le début et la fin hâtive de chaque tâche ainsi que le début et la fin au plus tard de chaque tâche. L'algorithme retourne aussi le nombre d'itérations effectuées.

Note finale du chapitre

Cette recherche ne propose pas de modification à la modélisation mathématique de Tissier (2008). Ce dernier avait mis à jour le modèle de Wiley et al (1998), qui est encore bien représentatif de la réalité. La contribution majeure est la mise en œuvre de l'algorithme tabou dans le modèle mathématique afin de créer un algorithme hybride. Le

but de l'algorithme tabou est de donner au modèle plus de flexibilité, rapidité et capacité.

Capacité

La taille du programme de projet varie énormément. L'algorithme tabou permet d'augmenter la capacité mathématique du modèle, c'est-à-dire de traiter des programmes de projets avec plus de tâches, car il relaxe certaines contraintes lors de la recherche d'activités à accélérer. De plus, l'algorithme permet de traiter plus de tâches à cause de l'efficience de ce dernier, puisqu'il s'agit d'une heuristique. Cependant, l'algorithme peut ne pas trouver la solution optimale, mais plutôt une solution quasi optimale. Dans un domaine comme la gestion multi-projets, plusieurs imprévus peuvent arriver. Des délais sont monnaie courante, alors un échange de précision pour de la capacité et de l'efficience est justifiable.

Rapidité

Une autre qualité de l'algorithme est sa rapidité d'exécution et sa rapidité à trouver une solution acceptable. Cette rapidité vient du fait que l'algorithme ne testera pas toutes les solutions possibles. En effet, il évolue dans le temps en estimant quels sont les changements, ou accélérations de tâches, qui ont le plus de chance d'améliorer la solution existante. De plus, il écarte les changements qui sont considérés comme des

mauvaises options en les classant comme tabou. De cette façon, l'algorithme ne perd pas de temps à explorer des solutions qui ne semblent pas optimales. Finalement, il gagne en rapidité et en capacité grâce à la flexibilité qui est donnée au gestionnaire de programmes.

Flexibilité

L'algorithme a été créé de façon à être adaptable au besoin du gestionnaire de multi-projets. Ces besoins peuvent varier en fonction de la grosseur (nombre de tâches) et de l'objectif à atteindre (en nombre de jours à accélérer). Effectivement, puisque le gestionnaire de projets peut préciser le nombre de jours à accélérer afin d'assurer le succès du programme, l'algorithme ne perd pas de temps à trouver des solutions une fois que cet objectif est atteint. Le gestionnaire de multi-projets peut aussi décider du nombre d'itérations nécessaires, alors il peut volontairement sacrifier un peu de précision afin de trouver une réponse à un problème de très grande envergure

Finalement, cet algorithme hybride répond au problème d'accélération de multi-projets et d'optimisation. Il sacrifie un peu de précision afin de gagner de la flexibilité, de la rapidité et de la capacité, ce qui fait en sorte qu'il répond mieux au besoin des gestionnaires de programmes avec la taille réelle des projets de nos jours.

Chapitre 4 : Application de l'algorithme sur une étude de cas

Lors de la planification d'un programme de projets, le gestionnaire aura des objectifs dictés, soit par le marché, soit par son entreprise. Lors de sa première planification, si les objectifs de durée et de coût ne sont pas atteints, le gestionnaire peut utiliser cet algorithme afin d'accélérer et d'optimiser le programme de projet.

Dans ce chapitre, l'algorithme hybride est testé sur une étude de cas. Afin de pouvoir comparer les résultats avec ceux de Tissier (2008), le même programme de projets est utilisé. De cette façon, nous prouverons l'amélioration de cet outil.

Deux types de données sont recueillis. Les données reliées à l'efficacité sont le coût et la durée du programme. Ils sont recueillis dans le but de comparer les deux algorithmes et sont particulièrement importants pour le gestionnaire de projets. La donnée reliée à l'efficience de l'algorithme est le nombre d'itérations, elle est particulièrement utile afin de comparer la rapidité d'exécution.

Le chapitre est divisé en trois parties. Premièrement, il pose le problème et explique chacun des projets. Deuxièmement, il présente les résultats obtenus lors de différentes itérations. Finalement, il fournit une analyse des résultats et compare ses derniers avec ceux de Tissier (2008).

4.1 Présentation du problème

Ce programme de projets est constitué de trois projets différents. Il ne comprend pas d'information reliée à la nature de la tâche, mais seulement celle nécessaire à l'accélération de projets. L'unité de temps est le jour ouvrable et l'unité de coût est le dollar canadien. Il n'y aura donc pas de discrimination par rapport à la nature des tâches, car il n'y a pas de priorisation pour cet aspect.

Projet A

Le projet A est tiré d'Azondékon (2002). La première table présente les détails reliés aux activités de ce dernier et un diagramme PERT représente ce projet dans ce stade original.

Table 4.1

Activités du projet A

Nom	Activités (i, j)	Prédécesseurs immédiats	Durée normale T_{ij} (en jours)	Possibilité d'accélérer M_{ij} (en jours)	Possibilité de décélérer N_{ij} (en jours)
A	1, 2	-	2	1	0
B	2, 3	A	7	1	0
C	2, 4	A	4	0	2
D	2, 5	A	2	0	4
E	3, 6	B	4	0	1
F	5, 6	B, C, D	6	1	0
G	3, 24	D	3	0	2
H	6, 24	E, F	2	1	0

La table 4.2 montre les caractéristiques financières du projet A. Les frais généraux de ce projet sont de 100 $ par jour. Le coût pour la durée initiale est de 57 400 $.

Table 4.2

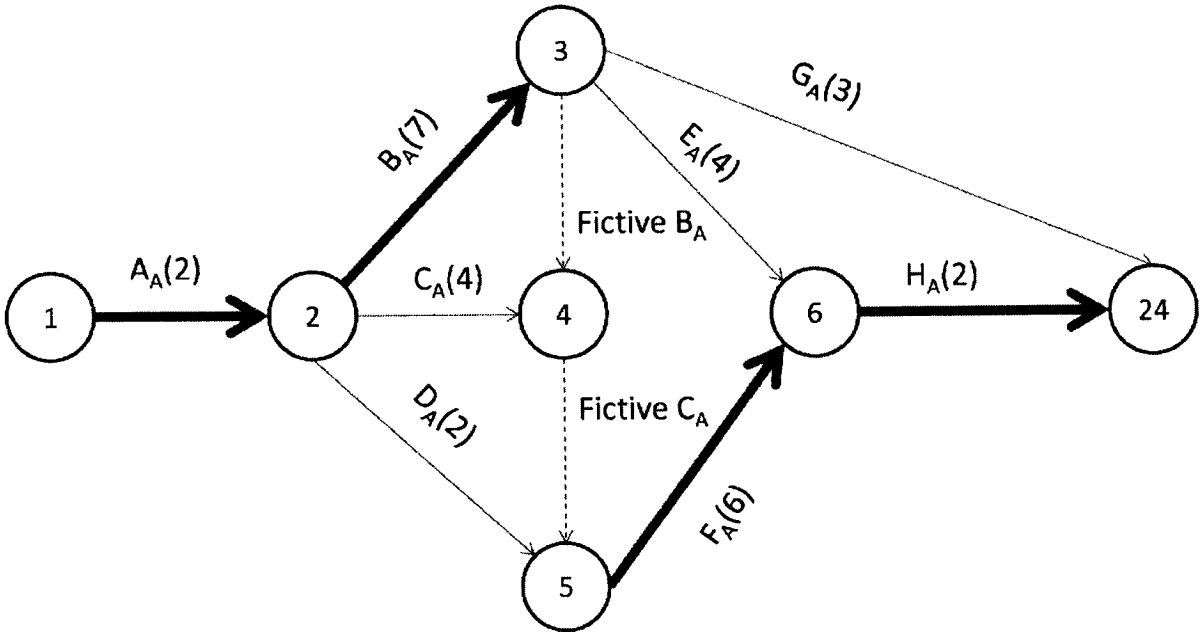
Caractéristiques financières du projet A

Nom	Activités (i, j)	Coût sur la durée normale D_{ij} (en $)	Coût par unité de temps accélérée K_{ij} (en $ / jour)	Coût par unité de temps décélérée A_{ij} (en $ / jour)
A	1, 2	3 000	1 500	-
B	2, 3	10 800	1 600	-
C	2, 4	4 800	-	1 200
D	2, 5	2 400	-	1 200
E	3, 6	6 000	-	1 500
F	5, 6	16 000	2 700	-
G	3, 24	2 400	-	800
H	6, 24	12 000	6 000	-

Le diagramme de PERT est présenté à la figure 4.1. Le projet A comprend huit tâches. Avec le diagramme de PERT, on observe que le chemin critique est de 17 jours. Il est possible d'accélérer le projet de 4 jours afin d'obtenir une durée de 13 jours. Par contre, comme nous allons voir dans les résultats, l'algorithme ne choisira pas d'accélérer ce projet au maximum. Toutefois, il sera beaucoup décéléré.

Figure 4.1

Diagramme PERT du projet A



Projet B

Le projet B est tiré d'Islam (2004). La première table représente les caractéristiques reliées aux activités, la deuxième table représente les caractéristiques financières et la figure représente le diagramme PERT.

Table 4.3

Activité du projet B

Nom	Activités (i, j)	Prédécesseurs immédiats	Durée normale T_{ij} (en jours)	Possibilité d'accélérer M_{ij} (en jours)	Possibilité de décélérer N_{ij} (en jours)
A	7, 8	-	3	1	0
B	8, 10	A	8	2	0
C	8, 9	A	3	0	1
D	9, 11	C	1	0	1
E	10, 12	B	3	0	1
F	11, 13	B, D	5	2	0
G	9, 13	C	2	0	2
H	13, 12	G, F	3	2	0
I	12, 24	E, H	2	1	0

La table 4.2 montre les caractéristiques financières du projet B. Les frais généraux de ce projet sont de 150 $ par jour. Le coût pour la durée initiale est de 76 200 $.

Table 4.4

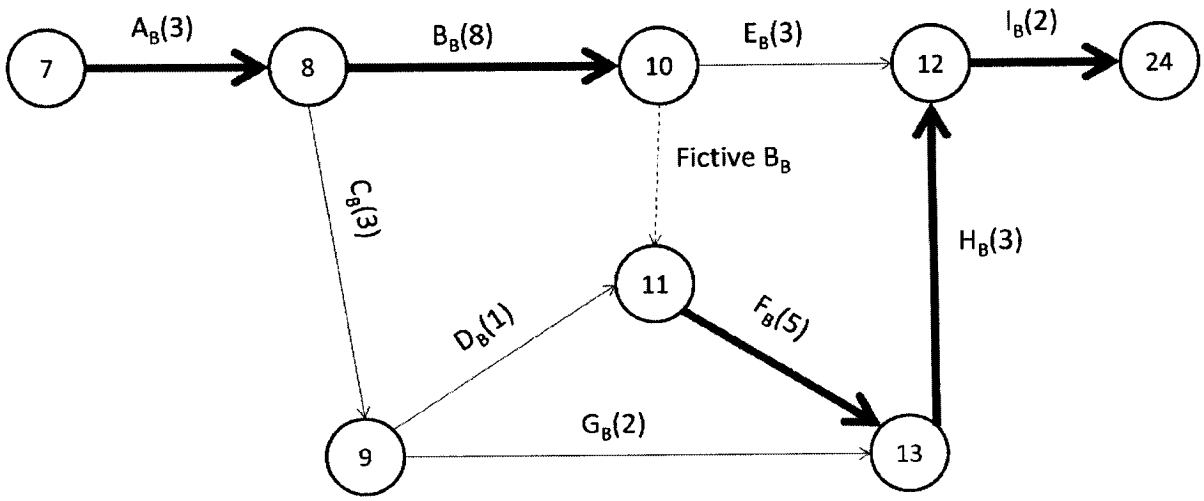
Caractéristiques financières du projet B

Nom	Activités (i, j)	Coût sur la durée normale D_{ij} (en $)	Coût par unité de temps accélérée K_{ij} (en $ / jour)	Coût par unité de temps décélérée A_{ij} (en $ / jour)
A	7, 8	16 000	5 400	-
B	8, 10	20 000	2 500	-
C	8, 9	4 000	-	1 400
D	9, 11	1 000	-	1 000
E	10, 12	12 000	-	4 000
F	11, 13	20 000	4 000	-
G	9, 13	1 000	-	500
H	13, 12	1 200	400	-
I	12, 24	1 000	500	-

Le diagramme de PERT du projet B est présenté à la figure 4.2. Ce projet comprend neuf tâches et son chemin critique est de 21 jours. Le projet B peut être accéléré de huit jours afin de raccourcir son chemin critique à treize jours. Encore une fois, l'algorithme ne choisira pas d'accélérer ce projet au maximum.

Figure 4.2

Diagramme de PERT du projet B



Projet C

Le projet C est tiré de Meredith et Mantel (2003). La première table représente les caractéristiques reliées aux activités, la deuxième table représente les caractéristiques financières et la figure représente le diagramme PERT.

Table 4.5

Activités du projet C

Nom	Activités (i, j)	Prédécesseurs immédiats	Durée normale T_{ij} (en jours)	Possibilité d'accélérer M_{ij} (en jours)	Possibilité de décélérer N_{ij} (en jours)
A	14, 15	-	1	0	0
B	15, 16	A	14	4	0
C	16, 17	B	5	0	5
D	16, 18	B	12	2	0
E	17, 20	C, D	5	2	0
F	16, 19	B	8	0	7
G	20, 21	E, F	5	0	0
H	21, 22	G	7	2	0
I	22, 23	H	2	1	0
J	23, 24	I	3	1	0

La table 4.5 montre les caractéristiques financières du projet C. Les frais généraux de ce projet sont de 200 $ par jour. Le coût pour la durée initiale est de 79 400 $.

Table 4.6

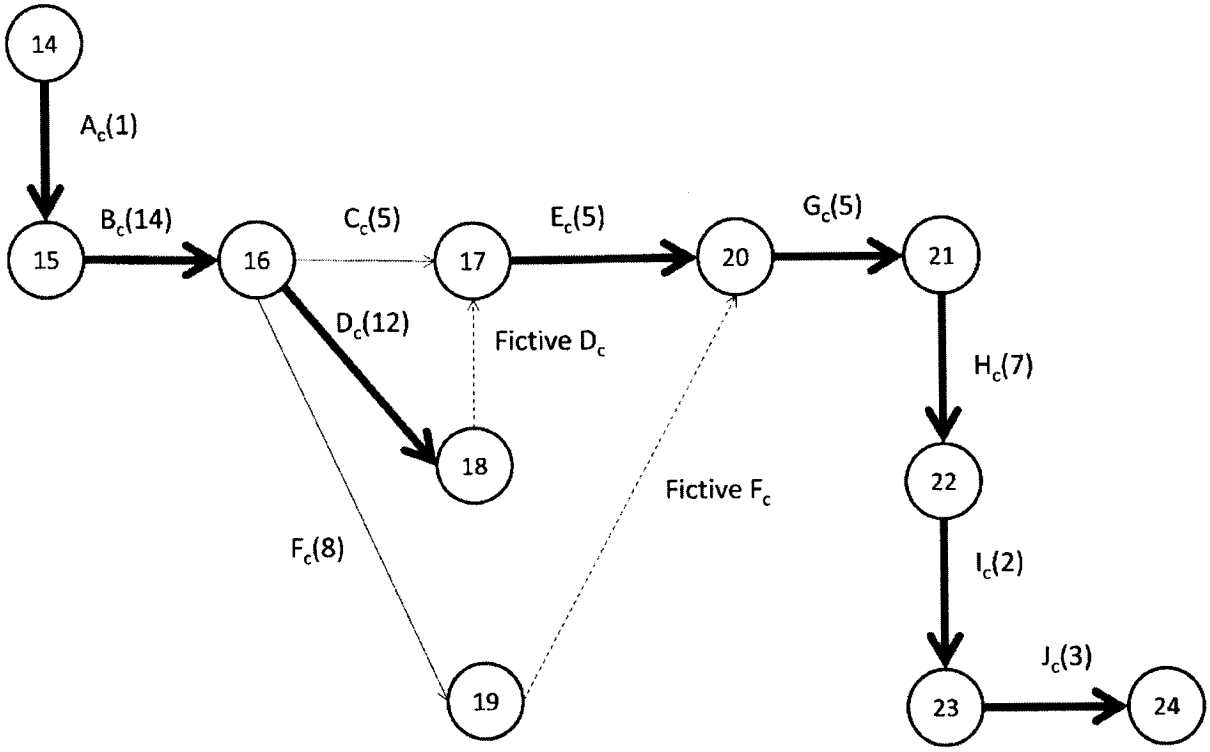
Caractéristiques financières du projet C

Nom	Activités (i, j)	Coût sur la durée normale D_{ij} (en $)	Coût par unité de temps accélérée K_{ij} (en $ / jour)	Coût par unité de temps décélérée A_{ij} (en $ / jour)
A	14, 15	600	-	-
B	15, 16	13 300	1 000	-
C	16, 17	5 600	-	1 200
D	16, 18	14 900	1 300	-
E	17, 20	4 800	1 000	-
F	16, 19	4 100	-	500
G	20, 21	19 600	-	-
H	21, 22	10 700	1 600	-
I	22, 23	1 400	700	-
J	23, 24	4 400	1 500	-

Le diagramme de PERT du projet C est présenté à la figure 4.3. Ce projet est constitué de dix tâches et son chemin critique est de 49 jours. Le projet C peut être accéléré de douze jours afin de raccourcir son chemin critique à 37 jours. Puisque le projet C est le plus imposant, c'est lui qui sera le plus accéléré.

Figure 4.3

Diagramme de PERT du projet C



Détail du programme

Les détails financiers reliés à l'ensemble du programme sont présentés dans la table 4.7. Celle-ci montre les budgets d'accélération permis de chaque projet ainsi que celui du programme. L'objectif de durée de 37 jours est aussi fixé pour le programme. Les bonus d'avancement sont appliqués à la date de fin du programme par rapport à la durée initialement planifiée.

Table 4.7

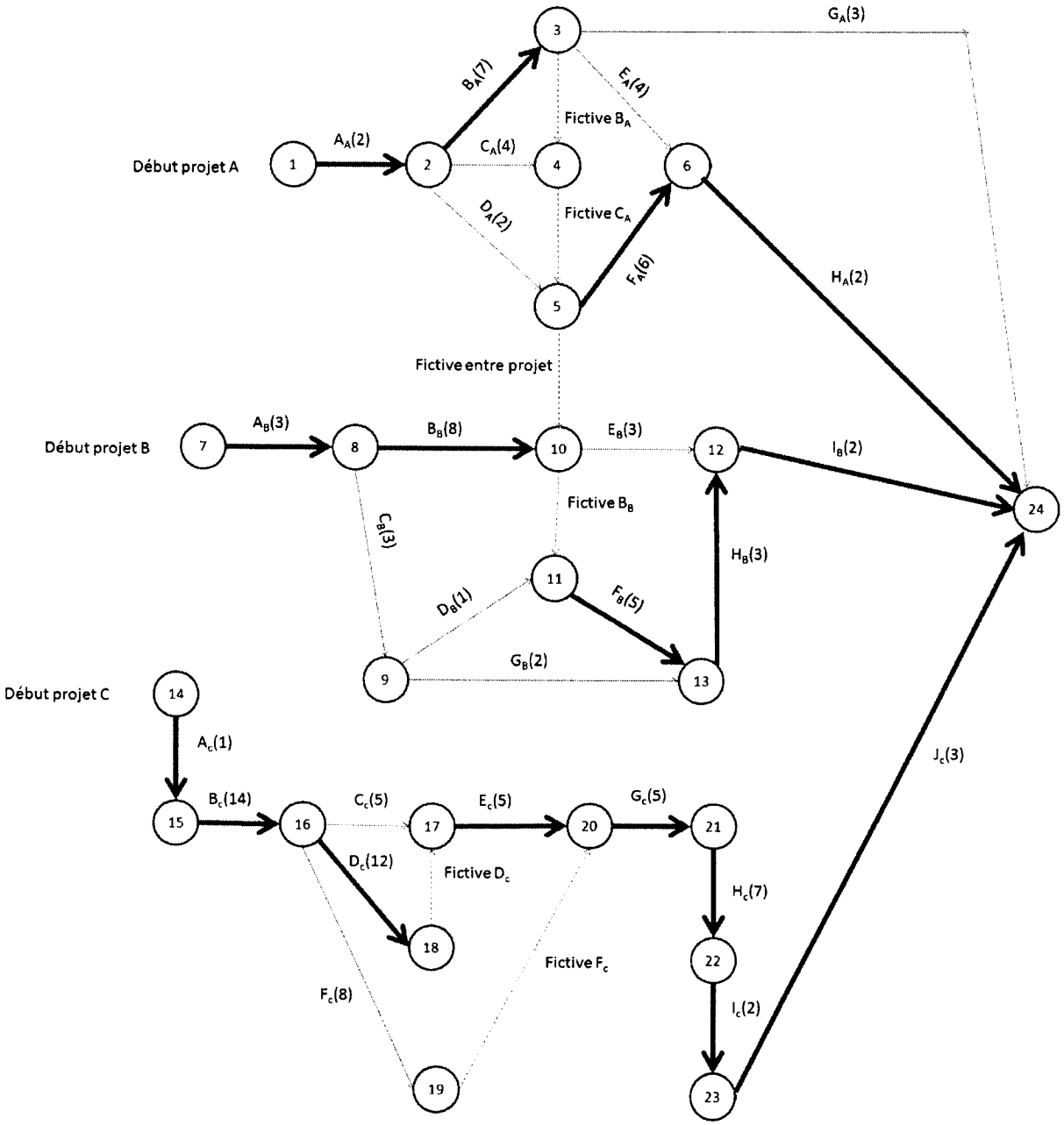
Finance du programme

Paramètres	Valeurs
Budget projet A	59 100 $
Budget projet B	79 350 $
Budget projet C	89 200 $
Budget programme	227 650 $
Budget accéléré projet A	70 500 $
Budget accéléré projet B	97 850 $
Budget accéléré projet C	100 800 $
Budget accéléré programme	269 150 $
Bonus d'avancement par jour (R)	400 $/jour
Durée initiale du programme (T)	49 jours
Objectif de durée du programme (TCD)	37 jours
Limite maximale du coût d'accélération du programme (P)	45 500 $
Frais généraux projet A (OH_1)	100 $/jour
Frais généraux projet B (OH_2)	150 $/jour
Frais généraux projet C (OH_3)	200 $/jour

La figure 4.4 représente le diagramme PERT du programme de projets. Le programme est constitué de 27 tâches et a une durée totale planifiée de 49 jours. Les trois projets peuvent être accélérés de plusieurs jours chacun, mais l'algorithme doit choisir les meilleures tâches à accélérer en fonction du budget. L'algorithme va toutefois tenter de décélérer les tâches au maximum, sauf si elles deviennent gênantes au chemin critique.

Figure 4.4

Diagramme PERT du programme



La table 4.8 représente un sommaire financier comparatif des projets. Le projet C est le plus dispendieux, en considérant les frais généraux, tandis que le projet A est le moins dispendieux.

Table 4.8

Sommaire financier du programme du programme original

		Projet A	Projet B	Project C	Programme
Original	Nombre de jours	17	21	49	49
	Coûts	57 400	76 200	79 400	213 000
	Frais généraux	1 700	3 150	9 800	14 650
	Bonus	0	0	0	0
	Total	59 100	79 350	89 200	227 650

4.2 Présentation des résultats

Les résultats ont été recueillis lors de différentes itérations. Un sommaire des résultats est présenté au début du chapitre afin de mettre en perspective les résultats finaux avec le programme de multi projet initialement planifié. Ensuite, les résultats finaux sont montrés en détail en ce qui concerne les activités accélérées et décélérés, le chemin critique, le sommaire financier du programme ainsi qu'une comparaison avec le programme initial. De plus, afin de suivre le cheminement logique de l'algorithme, les

itérations 1, 2 et 10 sont brièvement présentées dans l'annexe B afin d'augmenter la validité des résultats.

La dernière partie de ce chapitre effectue une comparaison de l'algorithme hybride avec celle de Tissier (2008) sur le plan d'efficience budgétaire et sur la rapidité d'exécution.

Table 4.9

Sommaire des résultats

		Projet A	**Projet B**	**Project C**	**Programme**
Original	Nombre de jours	17	21	49	49
	Coûts	59 100	79 350	89 200	227 650
Solution finale	Nombre de jours	16	18	37	37
	Coûts	54 200	72 400	92 500	213 700

Le tableau du sommaire des résultats montre que la solution finale trouvée accélère le programme de 12 jours et réduit les coûts de 13 950 $, soit 6,13 %.

Résultat détaillé de l'itération 30 (réponse finale)

L'itération 30 est la réponse finale pour ce programme. La décision d'arrêter à cette itération a été influencée par deux facteurs. Premièrement, l'objectif de durée du programme, qui est de 37 jours, a été atteint. Deuxièmement, toutes les contraintes sont respectées. Nous aurions pu laisser l'algorithme continuer pour plusieurs autres itérations, mais la solution ne change pratiquement plus, c'est-à-dire que l'algorithme perçoit cette réponse comme quasi optimale. En effet, les résultats parlent d'eux-mêmes. La table 4.10 montre les résultats d'accélération et de décélération des activités. On observe que les autres activités qui avaient temporairement été décélérées ([11,13] à l'itération 10) afin de débloquer l'algorithme ont été accélérées de nouveau afin de changer de voisinage immédiat. De cette façon, une nouvelle solution est trouvée afin de répondre à toutes les contraintes.

Table 4.10

Possibilités restantes à accélérer et à décélérer les activités

Activités	Jours restant à accélérer	Jours accélérés	Jours décélérés
1, 2	0	1	0
2, 3	1	0	0
2, 4	0	0	0
2, 5	0	0	4
3, 6	0	0	1
5, 6	1	0	0
3, 24	0	0	0
6, 24	1	0	0
7, 8	1	0	0
8, 10	2	0	0
8, 9	0	0	1
9, 11	0	0	1
10, 12	0	0	1
11, 13	2	0	0
9, 13	0	0	2
13, 12	0	2	0
12, 24	0	1	0
14, 15	0	0	0
15, 16	0	4	0
16, 17	0	0	5
16, 18	0	2	0
17, 20	0	2	0
16, 19	0	0	5
20, 21	0	0	0
21, 22	0	2	0
22, 23	0	1	0
23, 24	0	1	0

La table 4.11 représente le chemin critique de la solution. On observe que la majorité des activités sont sur le chemin critique. Ceci est une autre preuve que la

solution à l'itération 30 est optimisée. Au total, il n'y a que sept tâches sur vingt-sept qui ne sont pas sur le chemin critique.

Table 4.11

Résultat de la solution finale

Activités	Commencement au plus tôt	Fin au plus tôt	Commencement au plus tard	Fin au plus tard	Chemin critique = 1
1, 2	0	1	0	1	1
2, 3	1	8	1	8	1
2, 4	1	7	2	8	0
2, 5	1	7	2	8	0
3, 6	8	13	9	14	0
5, 6	8	14	8	14	1
3, 24	8	11	8	11	1
6, 24	14	16	14	16	1
7, 8	0	3	0	3	1
8, 10	3	11	3	11	1
8, 9	3	7	5	9	0
9, 11	7	9	9	11	0
10, 12	11	15	13	17	0
11, 13	11	16	11	16	1
9, 13	7	11	12	16	0
13, 12	16	17	16	17	1
12, 24	17	18	17	18	1
14, 15	0	1	0	1	1
15, 16	1	11	1	11	1
16, 17	11	21	11	21	1
16, 18	11	21	11	21	1
17, 20	21	24	21	24	1
16, 19	11	24	11	24	1
20, 21	24	29	24	29	1
21, 22	29	34	29	34	1
22, 23	34	35	34	35	1
23, 24	35	37	35	37	1

La table 4.12 montre le sommaire financier du programme de projet de la solution finale. De plus, il présente une comparaison avec la solution initiale. Le coût final du programme est de 205 700 $, ce qui représente une baisse de 7 300 $, ou de 3,43 %. Le coût total du programme incluant les frais généraux et les bonus est de 213 700 $, ce qui représente une baisse de 13 900 $, ou de 6,13 %, par rapport au programme initial. Il est important de noter l'optimisation effectuée par l'algorithme hybride. Le projet est non seulement raccourci de plusieurs jours, mais le coût est aussi plus petit. Ceci démontre qu'une bonne planification permet de profiter des bonus tout en payant moins de frais généraux. Un balancement des tâches est aussi important; il faut avoir le plus de tâches possible sur le chemin critique afin de maximiser son temps et ses ressources.

Table 4.12

Sommaire financier du programme de la solution finale

		Projet A	Projet B	Project C	Programme	Écart en $	Écart en %
Original	Nombre de jours	17	21	49	49		
	Coûts	57 400	76 200	79 400	213 000		
	Frais généraux	1 700	3 150	9 800	14 650		
	Bonus	-	-	-	0		
	Total	59 100	79 350	89 200	227 650		
Solution finale	Nombre de jours	16	18	37	37		
	Coûts	52 600	69 700	84 900	205 700	7300	3,43
	Frais généraux	1 600	2 700	7 600	12 000		
	Bonus	-	-	-	4 000		
	Total	54 200	72 400	92 500	213 700	13 950	6,13

4.3 Comparaison des résultats avec Tissier (2008)

La table 4.13 offre une comparaison des résultats de l'algorithme hybride avec la solution de l'optimisation de Tissier (2008). Les deux solutions obtiennent une durée finale de 37 jours, qui est aussi l'objectif général de l'optimisation déterminé par le gestionnaire. En ce qui a trait à la comparaison de l'accélération de la durée, les deux algorithmes sont égaux. Pour ce qui est des coûts, l'algorithme hybride trouve une solution de 17 240 $, ou 7,73 %, soit une solution moins chère que l'optimisation de

Tissier (2008). Pour les coûts totaux, incluant les frais généraux et les bonus, l'algorithme hybride trouve une solution moins chère de 15 890 $, ou 6,92 %. Pour ce qui est des itérations, l'algorithme hybride en a fait 30 contre 302 pour le modèle de Tissier (2008).

Table 4.13

Comparaison des résultats avec Tissier (2008)

		Projet A	Projet B	Project C	Programme	Écart en $	Écart en %
Tissier (2008)	Nombre de jours	15	17	37			
	Coûts	57 440	76 250	89 250	222 940		
	Frais généraux	1 500	2 550	7 400	11 450		
	Bonus				4 800		
	Total	58 940	78 800	91 850	229 590		
Solution finale	Nombre de jours	16	18	37	37		
	Coûts	52 600	69 700	84 900	205 700	17 240	7,73
	Frais généraux	1 600	2 700	7 600	12 000		
	Bonus		-	-	4 000		
	Total	54 200	72 400	92 500	213 700	15 890	6,92

Rapidité, capacité, flexibilité

Les résultats précédents démontrent bien la haute performance de l'algorithme hybride. En atteignant l'objectif d'accélération de 37 jours pour le programme, l'algorithme trouve une solution 6,92 % moins chère qu'avec le modèle de Tissier (2008). Ceci démontre un gain en efficience considérable. La principale qualité de l'algorithme est toutefois sa rapidité d'exécution. Alors que le modèle de Tissier (2008) trouve sa solution en 302 itérations, l'algorithme hybride la trouve en 30, ce qui a aussi un impact sur la capacité. Puisque la capacité maximale de traitement d'activité de l'algorithme n'est pas testée dans le cadre de cette recherche, on ne peut que stipuler sur celle-ci. Dans un problème NP-Hard, la difficulté se situe dans la quantité de solutions possibles à explorer en raison du grand nombre d'activités et de contraintes. Trente itérations sont significativement plus petites que 302 itérations, ce qui laisse croire qu'il y a un potentiel de capacité plus grand pour le traitement de données.

Quelques raisons expliquent la performance de l'algorithme hybride. Premièrement, l'algorithme est très efficace pour trouver les tâches non critiques ainsi que pour les décélérer afin d'augmenter le budget du programme et concentrer le budget sur les activités critiques, peu importe le projet. Ceci est un des bénéfices des programmes de multi-projets; les ressources sont constamment redistribuées aux projets les plus critiques afin d'optimiser le tout.

Note finale du chapitre

Les résultats de l'étude de cas permettent de qualifier l'algorithme comme efficace et 6,92 % plus efficient par rapport au modèle mathématique de Tissier (2008) en ce qui a trait au budget. L'algorithme hybride a été programmé avec l'outil mathématique Matlab. L'étude de cas a été directement réalisée avec l'algorithme. L'interface utilisateur n'était pas jugée essentielle afin de prouver le concept d'algorithme hybride. Une interface utilisateur sera nécessaire afin de commercialiser l'algorithme pour faciliter l'utilisation par les gestionnaires de programme de projets. Aussi, l'algorithme hybride comporte seulement le modèle d'accélération, le modèle d'allocation des ressources ne fait pas partie du cadre de cette recherche.

CONCLUSION

Le but de cette recherche était de développer un outil pour aider les gestionnaires de programme de projets à optimiser ces derniers. L'optimisation vise à réduire la durée du programme tout en minimisant les coûts. La principale optimisation est effectuée dans l'accélération de programme.

L'outil développé est un algorithme hybride. Nous avons repris le modèle mathématique d'accélération de programme de Tissier (2008), qui utilisait la méthode de décomposition Dantzig-Wolfe pour résoudre le problème à l'aide de programmation linéaire. À ce modèle mathématique, nous avons intégré un algorithme tabou. L'algorithme tabou est une heuristique utilisée pour trouver des solutions quasi optimales au problème NP-Hard. L'algorithme tabou commence par trouver une solution initiale valide, qui répond à toutes les contraintes et évolue ensuite dans le temps afin d'améliorer sa solution. Les options explorées par l'algorithme tabou, qui n'améliorent pas la solution, sont écartées et placées sur une liste dite tabou, afin de minimiser les solutions à explorer. Le modèle mathématique de Tissier (2008), avec l'algorithme tabou, forme un algorithme hybride. Cet algorithme hybride a été testé sur le même problème que Tissier (2008) afin de comparer les résultats.

L'algorithme hybride développé dans cette recherche a été appliqué dans une étude de cas. Cette étude de cas est constituée d'un programme comportant trois projets

d'une durée totale de 49 jours et coutant 227 650 $. La solution finale trouvée par Tissier (2008) est de 37 jours avec un budget de 222 940 $. La solution que nous avons trouvée est aussi de 37 jours, mais coute 213 700 $, ce qui représente une amélioration de 6,92 % par rapport à la solution de Tissier (2008). De plus, l'algorithme hybride trouve cette solution avec seulement 30 itérations, contre 302 itérations pour la solution de Tissier (2008).

L'étude de cas a donc prouvé l'efficacité de l'algorithme hybride développé lors de cette recherche. De plus, l'algorithme hybride est plus efficient sur les aspects de rapidité, de capacité et de flexibilité. En effet, les critères de l'algorithme peuvent être modifiés par le gestionnaire de programmes afin de mieux répondre à ses besoins. Puisque l'algorithme est une heuristique, elle n'explore pas nécessairement toutes les possibilités, mais seulement celles avec du potentiel. De cette façon, l'algorithme est beaucoup plus rapide à trouver une solution satisfaisante. On peut donc déduire qu'elle a une capacité à traiter plus de projets et de contraintes. Cependant, cette limite n'a pas été testée dans le cadre de cette recherche.

Notre contribution majeure se rapporte à l'intégration de l'algorithme tabou au modèle mathématique existant, formant alors l'algorithme hybride. Ce type d'algorithme n'avait jamais été testé dans le cadre de la gestion multi-projets. Les résultats de cette recherche démontrent que l'outil développé peut être utilisé par les gestionnaires de multi-projets afin d'optimiser leur projet.

Cette recherche a été effectuée dans certaines limites. Le type de ressources humaines n'a pas été considéré lors de l'optimisation, c'est-à-dire que l'algorithme hybride n'a pas été appliqué au second modèle de Tissier (2008). Aussi, tous les projets sont traités comme égaux, il n'y a donc pas de facteur d'importance dans l'accélération. L'algorithme hybride a été testé sur un programme de trois projets constitué de 27 tâches au total. Le nombre maximal de tâches traitables n'a pas été testé dans le cadre de cette recherche.

Cette recherche ouvre la porte à de futures améliorations et explorations. Une option serait d'incorporer le deuxième modèle de Tissier (2008) à l'algorithme hybride pour amener la composante des ressources humaines dans l'équation. Aussi, une des prémisses était que le temps et le coût sont liés par une fonction linéaire, ce qui constitue une approximation d'une fonction convexe. Une autre amélioration serait d'utiliser la fonction convexe telle quelle, ce qui ne devrait pas être un problème puisque l'algorithme hybride est une heuristique.

Index

Le domaine de gestion multi-projets utilise les concepts et les termes qui suivent. Ils sont utilisés par plusieurs chercheurs de ce domaine et peuvent susciter l'attention du lecteur.

- Algorithme tabou
- Algorithme hybride
- Gestion multi-projets
- Budget
- Programmation linéaire
- Décomposition Dantzig-wolfe
- Problème NP-Hard
- Accélération
- Décélération
- Réseau PERT
- Chemin critique
- Programme
- Matlab
- Contraintes
- Ressources humaines
- Optimisation

Bibliographie

Azaron, A., H. Katagiri, et al. (2006). "A multi-objective resource allocation problem in PERT networks." European Journal of Operational Research **172**(3): 838-854.

Azondékon, S. (2002). A Way to Deal With Time/Resource Trade-Off Problem. *Fifth International Conference of the International Research Network on Organizing by Project (IRNOP V).* Renesse, The Netherlands

Browning, T. R. and A. A. Yassine (2010). "Resource-constrained multi-project scheduling: Priority rule performance revisited." International Journal of Production Economics **126**(2): 212-228.

Coelho, J. and M. Vanhoucke (2011). "Multi-mode resource-constrained project scheduling using RCPSP and SAT solvers." European Journal of Operational Research **213**(1): 73-82.

Confessore, G., S. Giordani, et al. (2006). "A market-based multi-agent system model for decentralized multi-project scheduling." Annals of Operations Research **150**(1): 115-135.

Dantzig, G. B. and P. Wolfe (1960). "Decomposition principle for linear programs." Operations research **8**(1): 10.

Engwall, M. and A. Jerbrant (2003). "The resource allocation syndrome: the prime challenge of multi-project management?" International Journal of Project Management **21**.

Gendreau, M. (2002). An introduction to Tabu Search. Montréal, Université de Montréal.

Gonçalves, J. F., J. J. M. Mendes, et al. (2008). "A genetic algorithm for the resource constrained multi-project scheduling problem." European Journal of Operational Research **189**(3): 1171-1190.

Görög, M. (2011). "Translating single project management knowledge to project programs." Project Management Journal: n/a-n/a.

Islam, M. N., Rana, B. Md., Rafique, S., & Aziza, T. (2004). Crashing project time with least cost: A linear programming approach. *Journal of Business Research, Vol. 6*, 2004.

Jonas, D. (2010). "Empowering project portfolio managers: How management involvement impacts project portfolio management performance." International Journal of Project Management **28**(8): 818-831.

Jourdan, L., M. Basseur, et al. (2009). "Hybridizing exact methods and metaheuristics: A taxonomy." European Journal of Operational Research **199**(3): 620-629.

Krüger, D. and A. Scholl (2009). "A heuristic solution framework for the resource constrained (multi-)project scheduling problem with sequence-dependent transfer times." European Journal of Operational Research **197**(2): 492-508.

Montoya-Torres, J. R., E. Gutierrez-Franco, et al. (2010). "Project scheduling with limited resources using a genetic algorithm." International Journal of Project Management **28**(6): 619-628.

Nkubili, G., H. Kane, et al. (2011). Accélération de Projets dans un Contexte de Criticité de Ressources: Une approche basée sur l'algorithme Tabou. 1er Congrès International en Management et Gestion des projets. Gatineau (Québec), Canada, Université du québec en Outaouais.

Payne, J. H. (1995). "Management of multiple simultaneous projects: a state-of-the-art review." International Journal of Project Management **13**(3): 5.

Rabbani, M., S. M. T. Fatemi Ghomi, et al. (2007). "A new heuristic for resource-constrained project scheduling in stochastic networks using critical chain concept." European Journal of Operational Research **176**(2): 794-808.

Teller, J., B. N. Unger, et al. (2012). "Formalization of project portfolio management: The moderating role of project portfolio complexity." International Journal of Project Management **30**(5): 596-607.

Tissier, A. (2008). Accélération des projets et allocation des ressources : des enjeux de la gestion multi projet. Département des sciences administratives, Université du Québec en Outaouais. **Maitrise en Gestion de projet**.

Vasquez, M. (2001). An Hybrid Approach for the 0-1 Multi Knapsack Problem. 4th Methaheuristics International Conference Porto, Portugal.

Wiley, V. D., R. F. Deckro, et al. (1998). "Optimization analysis for design and planning of multi-project programs." European Journal of Operational Research **107**: 492-506.

Yaghootkar, K. and N. Gil (2012). "The effects of schedule-driven project management in multi-project environments." International Journal of Project Management **30**(1): 127-140.

Annexe A

Le code suivant a été programmé dans le logiciel Matlab. Il représente l'algorithme hybride et est directement appliqué au programme de projet défini dans l'étude de cas.

```
% Données spécifiques au programme A B C;
B=269150 ; % Budget du programme maximum (en $);
OH1=100 ; % Frais généraux par unité de temps (en $/jour) ;
OH2=150 ; % Frais généraux par unité de temps (en $/jour) ;
OH3=200 ; % Frais généraux par unité de temps (en $/jour) ;
R=400 ; % Bonus sur avancement de la date d'exécution du programme ;
T=49 ; % Chemin critique initial du programme ;
TCD=37 ; % Date désirée d'exécution du programme (en jours) ;
TA=17 ; % Durée désirée d'exécution du projet A (en jours) ;
TB=21 ; % Durée désirée d'exécution du projet B (en jours) ;
TC=49;  % Durée désirée d'exécution du projet C (en jours) ;
P=45500 ; % Limite maximale du cout d'accélération/décélération, sur le
programme(en $) ;

% Données spécifiques au projet A ;
BA=70500 ; % Budget du projet A (en $) ;
PA=11800 ; % Limite maximale du cout d'accélération/décélération, sur
le projet A(en $) ;
% Les durées normales (en jours) des activités (i, j) du projet A ;
T12=2 ; T23=7 ; T24=4 ; T25=2 ; T36=4 ; T56=6 ; T324=3 ; T624=2 ;
% Les coûts (en $) des activités (i, j) à leur durée normale, durant le
projet A ;
D12=3000 ; D23=10800 ; D24=4800 ; D25=2400 ; D36=6000 ; D56=16000 ;
D324=2400 ;
D624=12000 ;
% Les coûts (en $) par unité de temps, sur lesquelles l'activité (i, j)
est accélérée ;
K12=1500 ; K23=1600 ; K56=2700 ; K624=6000 ;
% Les limites supérieures (en jours) de yij ;
M12=1 ; M23=1 ; M56=1 ; M624=1 ;
% Les coûts (en $) par unité de temps, sur lesquelles l'activité (i, j)
est décélérée ;
```

```
A24=800 ; A25=400 ; A36=1200 ; A324=480 ;
% Les limites supérieures (en jours) de zij ;
N24=2 ; N25=4 ; N36=1 ; N324=2 ;

% Données spécifiques au projet B ;
BB=97850 ; % Budget du projet B (en $) ;
PB=19700 ; % Limite maximale du cout d'accélération/décélération, sur
le projet B(en $) ;
% Les durées normales (en jours) des activités (i, j) du projet B ;
T78=3 ; T810=8 ; T89=3 ; T911=1 ; T1012=3 ; T1113=5 ; T913=2 ; T1312=3
; T1224=2 ;
% Les coûts (en $) des activités (i, j) à leur durée normale, durant le
projet B ;
D78=16000 ; D810=20000 ; D89=4000 ; D911=1000 ; D1012=12000 ;
D1113=20000 ;
D913=1000 ; D1312=1200 ; D1224=1000 ;
% Les coûts (en $) par unité de temps, sur lesquelles l'activité (i, j)
est accélérée ;
K78=5400 ; K810=2500 ; K1113=4000 ; K1312=400 ; K1224=500 ;
% Les limites supérieures (en jours) de yij ;
M78=1 ; M810=2 ; M1113=2 ; M1312=2 ; M1224=1 ;
% Les coûts (en $) par unité de temps, sur lesquelles l'activité (i, j)
est décélérée ;
A89=1000 ; A911=500 ; A1012=3000 ; A913=250 ;
% Les limites supérieures (en jours) de zij ;
N89=1 ; N911=1 ; N1012=1 ; N913=2 ;

% Donnees specifiques au projet C ;
BC=100800 ; % Budget du projet C (en $) ;
PC=14000 ; % Limite maximale du cout d'accélération/décélération, sur
le projet C(en $) ;
% Les durées normales (en jours) des activités (i, j) du projet C ;
T1415=1 ; T1516=14 ; T1617=5 ; T1618=12 ; T1720=5 ; T1619=8 ; T2021=5 ;
T2122=7 ;
T2223=2 ; T2324=3 ;
% Les coûts (en $) des activités (i, j) à leur durée normale, durant le
projet C ;
D1415=600 ; D1516=13300 ; D1617=5600 ; D1618=14900 ; D1720=4800 ;
D1619=4100 ;
D2021=19600 ; D2122=10700 ; D2223=1400 ; D2324=4400 ;
% Les coûts (en $) par unité de temps, sur lesquelles l'activité (i, j)
est accélérée ;
K1516=1000 ; K1618=1300 ; K1720=1000 ; K2122=1600 ; K2223=700 ;
K2324=1500 ;
% Les limites supérieures (en jours) de yij ;
M1516=4 ; M1618=2 ; M1720=2 ; M2122=2 ; M2223=1 ; M2324=1 ;
% Les coûts (en $) par unité de temps, sur lesquelles l'activité (i, j)
est décélérée ;
A1617=560 ; A1619=270 ;
% Les limites supérieures (en jours) de zij ;
N1617=5 ; N1619=7 ;
% Activités fictives ;
```

```
T34=0; T45=0; T1011=0; T1817=0; T1920=0; T510=0 ;

net=[2 7 4 2 4 6 3 2 3 8 3 1 3 5 2 3 2 1 14 5 12 5 8 5 7 2 3]; %duré
normal des activité du program

%créé une activité 28 pour joindre tout les projet??? pas besoin,

%predecesseur projet a
pred=zeros (27);
pred (1,2)=1;
pred (1,3)=1;
pred (1,4)=1;
pred (2,5)=1;
pred (2,6)=1;
pred (3,6)=1;
pred (4,6)=1;
pred (2,7)=1;
pred (5,8)=1;
pred (6,8)=1;

%predecesseur projet b
pred (9,10)=1;
pred (9,11)=1;
pred (11,12)=1;
pred (10,13)=1;
pred (10,14)=1;
pred (12,14)=1;
pred (11,15)=1;
pred (14,16)=1;
pred (15,16)=1;
pred (13,17)=1;
pred (16,17)=1;

%predecesseur projet C
pred (18,19)=1;
pred (19,20)=1;
pred (19,21)=1;
pred (20,22)=1;
pred (21,22)=1;
pred (19,23)=1;
pred (22,24)=1;
pred (23,24)=1;
pred (24,25)=1;
pred (25,26)=1;
pred (26,27)=1;

apa=[1 1 0 0 0 1 0 1 1 2 0 0 0 2 0 2 1 0 4 0 2 2 0 0 2 1 1]; %nombre de
jour que chaque activité du program peut etre accéléré
dpa=[0 0 2 4 1 0 2 0 0 0 1 1 1 0 2 0 0 0 0 5 0 0 7 0 0 0 0]; %nombre de
jour que l on peur decelerer ON PEUT AUSSI DECELERER LES ACTIVITE
ACCELERE
```

```
cout_acc=[1500 1600 0 0 0 2700 0 6000 5400 2500 0 0 0 4000 0 400 500 0
1000 0 1300 1000 0 0 1600 700 1500]; %cout acceleration des activites
cout_dec=[0 0 1200 1200 1500 0 800 0 0 0 1400 1000 4000 0 500 0 0 0 0
1200 0 0 500 0 0 0 0]; %cout de deceleration des activites

%générer un chemin critique
[len,date]=criticalpath (net,pred);

%cout de base faux frais et activité normal
cbc=OH3*TC+D1415+D1516+D1617+D1618+D1720+D1619+D2021+D2122+D2223+D2324;
%cout de base du projet c
cba=OH1*TA+D12+D23+D24+D25+D36+D56+D324+D624;%cout de base du projet a
cbb=OH2*TB+D78+D810+D89+D911+D1012+D1113+D913+D1312+D1224;%cout de base
du projet c
cb=cbc+cba+cbb;%cout de base du program

%compteur pour les jours d'accélération
jour_acc=0;
caa=0;
cab=0;
cac=0;
jaa=0;
jab=0;
jac=0;
yy=0;
nouv_cout_acc=0;
%debut de l'algoritme tabou (va falloir checker la liste tabou)
i=0;
%trouver une solution initiale avec le system greedy.
%find critical activity to accelerate
hh=0;
gg=0;
zz=0;
xx=0;
uu=0;
oo=0;
pp=0;
w=0;
k=15;%Le choix du k représente le premier projet a etre accelerer dans
le prog.

while w< 10;



while k < length(net)+1; % pour k plus petit que le nombre d'activity
ERREUR PEUT ETRE QUE LON PEUT ACCELERER PLUS DE FOIS QUIL Y A DACTIVITY
(24 acceleration)
if date(0+k,5)==1 ; %trouve activité critique
    ac=k; %activité critique
   % k=length(net);%juste pour faire sortir de la boucle
```

```
   yy=yy+1; %DEBUG
if apa(ac)~=0;  % peut on accéléré l'activité critique?

    zz=zz+1; % debug check pour voir la loop

    %nouveau Cout d'accélération et décéleration


        if ac<9; % si la tache critique fait parti du projet A

            %test contrainte du budjet maximal d'accélération et
décélération
           if nouv_cout_acc+cout_acc(ac)<P && caa+cout_acc(ac)<PA  ;
%si le cout d'accélération est plus petit que le budjet d'accélération,

               %check budget total du projet
                if cb+nouv_cout_acc+cout_acc(ac)-R*(jour_acc+1)<B &&
cba+cout_acc(ac)<BA ;

                   caa=caa+ cout_acc(ac);    %calcul le cout acc du
projet a
                    jour_acc=jour_acc+1; %keep track of the
accelerated number of day in project A
                   cba=cba+cout_acc(ac);%calcul le cout du projet a
                    nouv_cout_acc=nouv_cout_acc+caa;%calcul le cout
accélération du programme

                    oo=oo+1;%DEBUG CHECK

                     net(ac)=net(ac)-1 ;% accelere activité par une
journée
                     apa(ac)=apa(ac)-1; % réduit de 1 le nombre de
jour que l'activité peut etre acceleré

                 else %l'activité ne peut-etre accéléré
                k=k+1;
                end
           else %l'activité ne peut-etre accéléré
            k=k+1;
           end
       end
       if 8<ac<18; % si la tache critique fait parti du projet A

            %test contrainte du budjet maximal d'accélération et
décélération
           if nouv_cout_acc+cout_acc(ac)<P  && cab+cout_acc(ac)<PB ;
%si le cout d'accélération est plus petit que le budjet d'accélération,

               %check budget total du projet
```

```
                if cb+nouv_cout_acc+cout_acc(ac)-R*(jour_acc+1)<B  &&
cbb+cout_acc(ac)<BB ;

                    cab=cab+cout_acc(ac);    %calcul le cout acc du
projet a
                    jour_acc= jour_acc+1; %keep track of the
accelerated number of day in project b
                     cbb=cbb+cout_acc(ac);%calcul le cout du projet b
                    nouv_cout_acc=nouv_cout_acc+cab; %calcul le cout
accélération du programme

                     net(ac)=net(ac)-1 ;% accelere activité par une
journée
                     apa(ac)=apa(ac)-1; % réduit de 1 le nombre de jour
que l'activité peut etre acceleré

                else %l'activité ne peut-etre accéléré
                 k=k+1;
                end
            else %l'activité ne peut-etre accéléré
             k=k+1;
            end
        end
        if 17<ac; % si la tache critique fait parti du projet A

            %test contrainte du budjet maximal d'accélération et
décélération
            if nouv_cout_acc+cout_acc(ac)<P  && cac+cout_acc(ac)<PC ;
%si le cout d'accélération est plus petit que le budjet d'accélération,
                %check budget total du projet
                if cb+nouv_cout_acc+cout_acc(ac)-R*(jour_acc+1)<B &&
cbc+cout_acc(ac)<BC;
                    cac=cac+cout_acc(ac);    %calcul le cout acc du
projet a
                     jour_acc=jour_acc+1; %keep track of the
accelerated number of day in project c
                     cbc=cbc+cout_acc(ac);%calcul le cout du projet c
                     nouv_cout_acc=nouv_cout_acc+cac; %calcul le cout
accélération du programme

                    net(ac)=net(ac)-1 ;% accelere activité par une
journée
                     apa(ac)=apa(ac)-1; % réduit de 1 le nombre de jour
que l'activité peut etre acceleré
                else %l'activité ne peut-etre accéléré
                 k=k+1;
                end
            else %l'activité ne peut-etre accéléré
            k=k+1;
            end
        end
```

```
 xx=xx+1; %DEBUG CHECK pour voir la loop

  %Calcul le nouveau cout du projet NEST PAS STORER ET DONC PERDU
 ncp=cb+nouv_cout_acc-R*(jour_acc);

else k=k+1;
end;

%Trouve le nouveau chemin critique
[len,date]=criticalpath (net,pred);

else k=k+1;
end;
end
if k==27 && hh<10; %le nombre d'itération de l'algoritme est limité
    k=1;
    hh=hh+1;
end;




%DECELERATION
%recherche du meilleur move possible dans le voisinage
%peut on décélérer l'activité non critique?
q=1;
rr=0;
tt=0;
while q < length(net)+1; %loop de deceleration
    if date(0+q,5)==0 ; %trouve activité non critique
        anc=q; %activité non critique

        rr=rr+1; %DEBUG CHECK

        if dpa (anc)~=0;  % peut on décélérer l'activité non critique?

            tt=tt+1; %DEBUG CHECK

             %calcul le cout acc du projet a, b,c et programme
        if anc<9; % si la tache critique fait parti du projet A
        caa=caa- cout_dec(anc);    %calcul le cout acc du projet a
         cba=cba-cout_dec(anc);%calcul le cout du projet a
        nouv_cout_acc=nouv_cout_acc-cout_dec(anc);%calcul le cout
accélération du programme
        end;
        if 8<anc<18; % si la tache critique fait parti du projet A
        cab=cab-cout_dec(anc);    %calcul le cout acc du projet a
         cbb=cbb-cout_dec(anc);%calcul le cout du projet b
```

```
        nouv_cout_acc=nouv_cout_acc-cout_dec(anc); %calcul le cout
accélération du programme
        end;
        if 17<anc; % si la tache critique fait parti du projet A
        cac=cac-cout_dec(anc);    %calcul le cout acc du projet a
         cbc=cbc-cout_dec(anc);%calcul le cout du projet c
       nouv_cout_acc=nouv_cout_acc-cout_dec(anc); %calcul le cout
accélération du programme
        end;

        net(anc)=net(anc)+1; %decelere activité par une journée
        dpa(anc)=dpa(anc)-1; %réduit de 1 le nombre de jour possible a
décélérer

        %Calcul le nouveau cout du projet NEST PAS STORER ET DONC PERDU
             ncp=cb+nouv_cout_acc-R*(jour_acc);

        %cb=cb+nouv_cout_acc-R*(jour_acc);

        else  q=q+1;
        end;

    else q=q+1; %prochaine acctivité non critique
    end;
    %Trouve le nouveau chemin critique
[len,date]=criticalpath (net,pred);

end;
w=w+1;
%k=1; %reset le compteur d'acceleration
end;

%store solution initiale
sol1_len=len;
sol1_date=date;
net1=net;

apa1=apa;
dpa1=dpa;
cout_acc1=cout_acc;
cout_dec1=cout_dec;

nouv_cout_acc1=nouv_cout_acc;
caa1=caa;
jour_acc1=jour_acc;
cba1=cba;
cab1=cab;
cbb1=cbb;
cac1=cac;
cbc1=cbc;
```

```
ncp1=ncp;



%fonction pour unjammer l'algoritme tabou quand elle tourne en rond

%batir la liste tabou

tabou=cout_acc; %créé la liste tabou

compteurtabou=0;

gb=0;
while gb< 1000; % Le nombre d'itération de l'algorithme tabou

    %compte les itération de tabou
    compteurtabou=compteurtabou+1;

  g=1;
while g < length(apa)+1;
l=1;
    if apa(g)==0;
   tabou(g)=99999; %rend tabou les activités ne pouvant etre accéléré
    if cout_acc(g)==0;
        tabou(g)=99999;
    end
    end
    g=g+1;
end

s=1; %valeur initiale de s

               %check si tout les moves sont tabou
    nm=min(tabou); % trouve le plus petit chiffre de tabou
    if nm==99999;
   l=15; %terminate program because no more activity to accelerate.
    end



while l < 3; % on fait 2 iteration relaxer random et on check la
solution voir si elle est meilleur (24 acceleration)

   %fonction pour trouvé le prochain move random qui nest pas tabou
  %  vvv=1;
  % randomtabou=randsample(tabou,1);
   %while randomtabou==99999 && vvv<500;
    %    randomtabou=randsample(tabou,1);
     %   vvv=vvv+1;
   %end
```

```
   %ll=find(tabou==randomtabou); %sélectionne la position du chiffre
random
   %s=ll(1);

    nm=min(tabou); % trouve le plus petit chiffre de tabou
   ll=find(tabou==nm); %retourne la position de l'accélération la plus
cheap dans la matrice tabout
   s=ll(1); %si il y a plusieurs minimum dans tabou, on va prendre le
premier

   if l==1; %determine le premier move. le premier move (ss) est celui
qui doit etre tabou sil échou
       ss=s;
   end
        if s<9; % si la tache critique fait parti du projet A

            %l'algorithme tabou ne teste pas toute les contraintes,
elle
            %relaxe le probleme

            %test contrainte du budjet maximal d'accélération et
décélération
            if nouv_cout_acc+cout_acc(s)<P ; %si le cout d'accélération
est plus petit que le budjet d'accélération,

                %check budget total du projet
                 if cb+nouv_cout_acc+cout_acc(s)-R*(jour_acc+1)<B  ;

                    caa=caa+ cout_acc(s);    %calcul le cout acc du
projet a
                     jour_acc=jour_acc+1; %keep track of the
accelerated number of day in project A
                    cba=cba+cout_acc(s);%calcul le cout du projet a
                     nouv_cout_acc=nouv_cout_acc+caa;%calcul le cout
accélération du programme

                      net(s)=net(s)-1 ;% accelere activité par une
journée
                      apa(s)=apa(s)-1; % réduit de 1 le nombre de jour
que l'activité peut etre acceleré

                       if apa(s)==0;
                         tabou(s)=99999;
                       end
                 else %l'activité ne peut-etre accéléré
                tabou(s)=99999; %considéré tabou
                end
            else %l'activité ne peut-etre accéléré
             tabou(s)=99999; %considéré tabou
            end
        end
```

```
        if 8<s<18; % si la tache critique fait parti du projet b

             %test contrainte du budjet maximal d'accélération et
décélération
            if nouv_cout_acc+cout_acc(s)<P ; %si le cout d'accélération
est plus petit que le budjet d'accélération,

                %check budget total du projet
                if cb+nouv_cout_acc-R*(jour_acc+1)<B ; %+cout_acc(s)

                    cab=cab+cout_acc(s);    %calcul le cout acc du
projet a
                    jour_acc= jour_acc+1; %keep track of the
accelerated number of day in project b
                     cbb=cbb+cout_acc(s);%calcul le cout du projet b
                    nouv_cout_acc=nouv_cout_acc+cab; %calcul le cout
accélération du programme

                     net(s)=net(s)-1 ;% accelere activité par une
journée
                     apa(s)=apa(s)-1; % réduit de 1 le nombre de jour
que l'activité peut etre acceleré

                        if apa(s)==0;
                         tabou(s)=99999;
                       end
                else %l'activité ne peut-etre accéléré
                 tabou(s)=99999; %considéré tabou
                end
            else %l'activité ne peut-etre accéléré
            tabou(s)=99999; %considéré tabou
            end
        end
        if 17<s; % si la tache critique fait parti du projet A

            %test contrainte du budjet maximal d'accélération et
décélération
            if nouv_cout_acc+cout_acc(s)<P ; %si le cout d'accélération
est plus petit que le budjet d'accélération,
                %check budget total du projet
                if cb+nouv_cout_acc+cout_acc(s)-R*(jour_acc+1)< B ;
                    cac=cac+cout_acc(s);    %calcul le cout acc du
projet a
                     jour_acc=jour_acc+1; %keep track of the
accelerated number of day in project c
                     cbc=cbc+cout_acc(s);%calcul le cout du projet c
                     nouv_cout_acc=nouv_cout_acc+cac; %calcul le cout
accélération du programme

                    net(s)=net(s)-1 ;% accelere activité par une
journée
```

```
                    apa(s)=apa(s)-1; % réduit de 1 le nombre de jour
que l'activité peut etre acceleré
                        if apa(s)==0;
                         tabou(s)=99999;
                        end
                else %l'activité ne peut-etre accéléré
                 tabou(s)=99999; %considéré tabou
                end
            else %l'activité ne peut-etre accéléré
            tabou(s)=99999; %considéré tabou
            end
        end


  %Calcul le nouveau cout du projet NEST PAS STORER ET DONC PERDU
 ncp=cb+nouv_cout_acc-R*(jour_acc);

 l=l+1;
end;


%DECELERATION
%recherche du meilleur move possible dans le voisinage
%peut on décélérer l'activité non critique?
q=1;
rr=0;
tt=0;
while q < length(net)+1; %loop de deceleration
    if date(0+q,5)==0 ; %trouve activité non critique
        anc=q; %activité non critique

        rr=rr+1; %DEBUG CHECK

        if dpa (anc)~=0;  % peut on décélérer l'activité non critique?

            tt=tt+1; %DEBUG CHECK

             %calcul le cout acc du projet a, b,c et programme
        if anc<9; % si la tache critique fait parti du projet A
        caa=caa- cout_dec(anc);    %calcul le cout acc du projet a
         cba=cba-cout_dec(anc);%calcul le cout du projet a
        nouv_cout_acc=nouv_cout_acc-cout_dec(anc);%calcul le cout
accélération du programme
        end;
        if 8<anc<18; % si la tache critique fait parti du projet A
        cab=cab-cout_dec(anc);    %calcul le cout acc du projet a
         cbb=cbb-cout_dec(anc);%calcul le cout du projet b
        nouv_cout_acc=nouv_cout_acc-cout_dec(anc); %calcul le cout
accélération du programme
        end;
        if 17<anc; % si la tache critique fait parti du projet A
```

```
        cac=cac-cout_dec(anc);    %calcul le cout acc du projet a
         cbc=cbc-cout_dec(anc);%calcul le cout du projet c
       nouv_cout_acc=nouv_cout_acc-cout_dec(anc); %calcul le cout
accélération du programme
        end;

        net(anc)=net(anc)+1; %decelere activité par une journée
        dpa(anc)=dpa(anc)-1; %réduit de 1 le nombre de jour possible a
décélérer

        %Calcul le nouveau cout du projet NEST PAS STORER ET DONC PERDU
             ncp=cb+nouv_cout_acc-R*(jour_acc);

        %cb=cb+nouv_cout_acc-R*(jour_acc);

        else  q=q+1;
        end;

    else q=q+1; %prochaine acctivité non critique
    end;

end;


[len,date]=criticalpath (net,pred);%Trouve le nouveau chemin critique

net2=net; %solution tentative
sol2_len=len;
sol2_date=date;

%TEST SI LE MOUVE EST VALIDE
    if nouv_cout_acc+cout_acc(s)<P && caa+cout_acc(s)<PA &&
cb+nouv_cout_acc+cout_acc(s)-R*(jour_acc+1)<B && cba+cout_acc(s)<BA &&
nouv_cout_acc+cout_acc(s)<P  && cab+cout_acc(s)<PB &&
cb+nouv_cout_acc+cout_acc(s)-R*(jour_acc+1)<B  && cbb+cout_acc(s)<BB &&
nouv_cout_acc+cout_acc(s)<P  && cac+cout_acc(s)<PC &&
cb+nouv_cout_acc+cout_acc(s)-R*(jour_acc+1)<B &&  cbc+cout_acc(s)<BC;
        % Solution est valide
       if sol2_len<sol1_len; %Est-ce que la solution est meilleur que
la solution initiale


            %store la nouvelle solution
             sol1_len=sol2_len;
           sol1_date=sol2_date;
           net1=net2;

                apa1=apa;
                dpa1=dpa;
                cout_acc1=cout_acc;
```

```
                cout_dec1=cout_dec;

                nouv_cout_acc1=nouv_cout_acc;
                caa1=caa;
                jour_acc1=jour_acc;
                cba1=cba;
                cab1=cab;
                cbb1=cbb;
                cac1=cac;
                cbc1=cbc;

                ncp1=ncp;

               %Si la solution est ce que lon veut, on arrete
l'algorithme
                if net==37;
               gb=200;
                end

       else
            % LE MOVE N' EST PAS VALIDE
           %remet les activitées initiale et store le move en tabou
                net= net1;
               apa =apa1;
                dpa=dpa1;
                cout_acc=cout_acc1;
                cout_dec=cout_dec1;

                nouv_cout_acc=nouv_cout_acc1;
                caa=caa1;
                jour_acc=jour_acc1;
                cba=cba1;
                cab=cab1;
                cbb=cbb1;
                cac=cac1;
                cbc=cbc1;
                ncp=ncp1;

                tabou(ss)= 99999;  %METTRE LES MOVES EN TABOU!!!! met
le premier move en tabou

                net(s)=net(s)+1 ;% remet le net a loriginale
                     apa(s)=apa(s)+1; % remet apa a loriginal


       end

       else
            % LE MOVE N' EST PAS VALIDE
           %remet les activitées initiale et store le move en tabou
                net= net1;
```

```
                apa =apa1;
                 dpa=dpa1;
                 cout_acc=cout_acc1;
                 cout_dec=cout_dec1;

                 nouv_cout_acc=nouv_cout_acc1;
                 caa=caa1;
                 jour_acc=jour_acc1;
                 cba=cba1;
                 cab=cab1;
                 cbb=cbb1;
                 cac=cac1;
                 cbc=cbc1;
                 ncp=ncp1;

                 tabou(ss)= 99999;  %METTRE LES MOVES EN TABOU!!!! met
le premier move en tabou

                 net(s)=net(s)+1 ;% remet le net a loriginale
                      apa(s)=apa(s)+1; % remet apa a loriginal
    end

gb=gb+1;
end;

iii=1;
while iii<28;
    if apa(iii)<0;
    net(iii)=net(iii)+1;
    apa(iii)=0;
    end
    iii=iii+1;
end
[len,date]=criticalpath (net,pred);%Trouve le nouveau chemin critique
```

Annexe B

L'annexe B présente les résultats partiels de cette recherche. Il explique les résultats de l'itération 1, 2 et 10 afin d'augmenter la validité des résultats finaux.

Solution initiale (itération 1)

La solution initiale est particulièrement importante, car elle détermine la rapidité d'exécution de l'algorithme. En effet, plus elle est performante, plus elle sauvera du temps d'exécution plus tard. La solution initiale a réussi à accélérer le programme jusqu'à une durée de 38 jours. On observe dans la table du chemin critique de la solution initiale qu'il y 20 activités qui sont maintenant critiques sur un total de 27 activités, ce qui est beaucoup.

Table B1

Chemin critique de la solution initiale

Activités	Commencement au plus tôt	Fin au plus tôt	Commencement au plus tard	Fin au plus tard	Chemin critique = 1
1, 2	0	2	0	2	1
2, 3	2	9	2	9	1
2, 4	2	8	3	9	0
2, 5	2	8	3	9	0
3, 6	9	14	10	15	0
5, 6	9	15	9	15	1
3, 24	9	12	9	12	1
6, 24	15	17	15	17	1
7, 8	0	3	0	3	1
8, 10	3	11	3	11	1
8, 9	3	7	5	9	0
9, 11	7	9	9	11	0
10, 12	11	15	13	17	0
11, 13	11	16	11	16	1
9, 13	7	11	12	16	0
13, 12	16	17	16	17	1
12, 24	17	18	17	18	1
14, 15	0	1	0	1	1
15, 16	1	11	1	11	1
16, 17	11	21	11	21	1
16, 18	11	21	11	21	1
17, 20	21	24	21	24	1
16, 19	11	24	11	24	1
20, 21	24	29	24	29	1
21, 22	29	34	29	34	1
22, 23	34	35	34	35	1
23, 24	35	38	35	38	1

La table suivante démontre le résultat des activités accélérées et des activités décélérées. Nous pouvons observer que la solution initiale met beaucoup d'accent sur la décélération des activités. De cette façon, la solution initiale permet de maximiser l'économie d'argent sur les tâches non critiques et ensuite, peut la réinvestir dans les tâches critiques afin de raccourcir la durée totale du programme. Aussi, on peut voir que l'algorithme met l'accent sur l'accélération du projet C. Le projet C est le plus long et comporte plus de tâches sur le chemin critique.

Table B2

Activité accélérée et décéléré de la solution initiale

Activités	Jour restant à accélérer	Jour accéléré	Jour décéléré
1, 2	1	0	0
2, 3	1	0	0
2, 4	0	0	0
2, 5	0	0	4
3, 6	0	0	1
5, 6	1	0	0
3, 24	0	0	0
6, 24	1	0	0
7, 8	1	0	0
8, 10	2	0	0
8, 9	0	0	1
9, 11	0	0	1
10, 12	0	0	1
11, 13	2	0	0
9, 13	0	0	2
13, 12	0	2	0
12, 24	0	1	0
14, 15	0	0	0
15, 16	0	4	0
16, 17	0	0	5
16, 18	0	2	0
17, 20	0	2	0
16, 19	0	0	5
20, 21	0	0	0
21, 22	0	2	0
22, 23	0	1	0
23, 24	1	0	0

Un sommaire financier du programme, lors de la solution initiale, ainsi qu'une comparaison avec l'état financier du programme original (non optimisé) sont présentés dans la table B3. La solution initiale améliore la durée totale du programme jusqu'à

38 jours. Par contre, on n'atteint pas encore l'objectif de 37 jours. Il y a une optimisation au niveau du budget du programme, car le coût des projets diminue de 8 800 $, ce qui constitue une réduction de 4,13%. En considérant tous les coûts du programme, ce dernier économise 15 050 $ grâce au bonus d'avancement et des frais généraux réduits à cause de l'accélération. Il y a donc une amélioration budgétaire de 6,61 %. Il est à noter que le pourcentage correspond à l'économie d'argent sur le coût initial, multiplié par cent.

Table B3

Sommaire financier du programme lors de la solution initiale

		Projet A	Projet B	Project C	Programme	Écart en $	Écart en %
Original	Nombre de jours	17	21	49	49		
	Coûts	57 400	76 200	79 400	213 000		
	Frais généraux	1 700	3 150	9 800	14 650		
	Bonus	0	0	0	0		
	Total	59 100	79 350	89 200	227 650		
Solution initiale	Nombre de jours	17	18	38	38		
	Coûts	51 100	69 700	83 400	204 200	8 800	4,13
	Frais généraux	1 700	2 700	7 600	12 000		
	Bonus	-	-	-	3 600		
	Total	52 800	72 400	91 000	212 600	15 050	6,61

Itération 2

La deuxième itération est uniquement présentée dans le but de valider le processus logique de l'algorithme. Comme il a été mentionné auparavant, l'itération deux ne doit pas améliorer la durée du programme à cause de la nature de la solution initiale. Puisque la solution initiale est optimisée au maximum dans son entourage immédiat, la deuxième itération a comme tâche de changer de voisinage afin de redécouvrir de nouvelles possibilités. On valide le modèle en observant que la durée totale du programme reste à 38 jours (comme à la solution initiale). L'itération deux va toutefois faire une accélération qui brise les contraintes du programme. Donc, la solution présentée à l'itération deux n'est pas valide. Les contraintes brisées sont une nécessité afin de changer de voisinages et devront être reconfirmées plus tard. La table B4 démontre les activités accélérées pour cette itération. L'algorithme décélère temporairement l'activité (8,10) afin de pouvoir changer de voisinage.

Table B4

Activité accélérée pour l'itération deux

Activités	Jour restant à accélérer	Jour accéléré
1, 2	0	1
2, 3	1	0
2, 4	0	0
2, 5	0	0
3, 6	0	0
5, 6	1	0
3, 24	0	0
6, 24	1	0
7, 8	1	0
8, 10	3 (décéléré)	décéléré de 1
8, 9	0	0
9, 11	0	0
10, 12	0	0
11, 13	2	0
9, 13	0	0
13, 12	0	2
12, 24	0	1
14, 15	0	0
15, 16	0	4
16, 17	0	0
16, 18	0	2
17, 20	0	2
16, 19	0	0
20, 21	0	0
21, 22	0	2
22, 23	0	1
23, 24	1	0

Itération 10

L'itération 10 est la première à atteindre l'objectif d'un programme d'une durée de 37 jours. Toutefois, certaines des contraintes ne sont pas respectées et l'algorithme ne retiendra pas cette itération. Ce qui est intéressant est que l'algorithme comprend qu'il a enfreint des contraintes et revient sur ses pas. Les activités accélérées pour l'itération dix sont présentées à la table B5. L'algorithme décélère une activité temporairement (11,13) afin de sortir de son voisinage immédiat. Il est important de noter que l'algorithme a accéléré de nouveau l'activité (8,10) de l'itération 10 afin de gagner du temps. Ceci démontre bien le mécanisme de changement de voisinage ainsi que le prolongement temporaire du programme afin de mieux l'optimiser plus tard.

Table B5

Activité accélérée pour l'itération dix

Activités	Jour restant à accélérer	Jour accéléré
1, 2	0	1
2, 3	1	0
2, 4	0	0
2, 5	0	0
3, 6	0	0
5, 6	1	0
3, 24	0	0
6, 24	1	0
7, 8	1	0
8, 10	2	0
8, 9	0	0
9, 11	0	0
10, 12	0	0
11, 13	3	décéléré
9, 13	0	0
13, 12	0	2
12, 24	0	1
14, 15	0	0
15, 16	0	4
16, 17	0	0
16, 18	0	2
17, 20	0	2
16, 19	0	0
20, 21	0	0
21, 22	0	2
22, 23	0	1
23, 24	0	0