UNIVERSITÉ DE MONTRÉAL

ÉTUDE DE L'IMPACT DU CHEVAUCHEMENT SUR LES PERFORMANCES DE PROJETS COMPLEXES D'INGÉNIERIE

FRANÇOIS BERTHAUT

DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION

DU DIPLÔME DE PHILOSOPHAE DOCTOR

(GÉNIE INDUSTRIEL)

AOÛT 2015

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

ÉTUDE DE L'IMPACT DU CHEVAUCHEMENT SUR LES PERFORMANCES DE PROJETS COMPLEXES D'INGÉNIERIE

présentée par : BERTHAUT François

en vue de l'obtention du diplôme de : Philosophae Doctor

a été dûment acceptée par le jury d'examen constitué de :

M. BAPTISTE Pierre, Doctorat, président

M. PELLERIN Robert, Ph. D., membre et directeur de recherche

M. HAJJI Adnène, Ph. D., membre et codirecteur de recherche

M. BASSETTO Samuel, Doctorat, membre

M. GARDONI Mickaël, Doctorat, membre externe

DÉDICACE

À ma femme Marie-Élaine et à mes deux filles Charlotte et Florence

REMERCIEMENTS

Je souhaite remercier en premier lieu mon directeur de recherche, le Pr. Robert Pellerin, et mon co-directeur de recherche, le Pr. Adnène Hajji, pour la confiance qu'ils m'ont accordée durant les cinq années de mon doctorat. Sans leur disponibilité, leurs directives, leurs conseils et leur support, ces travaux n'auraient pu être menés à bien.

J'adresse aussi mes remerciements aux membres du jury, les professeurs Pierre Baptiste, Samuel Bassetto et Mickael Gardoni, qui me font l'honneur d'examiner ces travaux et d'en être les rapporteurs.

Je tiens également à remercie Nathalie Perrier, assistante de recherche dans la Chaire de recherche Jarislowsky / SNC-Lavalin en gestion de projets internationaux avec qui j'ai partagé mon bureau, pour son soutien quotidien et ses conseils.

Je souhaite remercier aussi les personnes et organismes qui m'ont soutenu financièrement durant mes travaux : Mr. Stephen A. Jarislowsky qui soutient personnellement la Chaire et à qui j'ai eu la formidable opportunité de présenter mes travaux, notre collaborateur industriel, la Fondation Polytechnique, le CRSNG pour la Bourse d'études supérieures du Canada Alexander-Graham Bell, et enfin le CIRRELT pour la Bourse d'excellence d'accueil au doctorat, la Bourse d'excellence de rédaction au doctorat et la Bourse de coopération Montréal-Québec.

Je souhaite exprimer ma gratitude à notre collaborateur industriel pour m'avoir offert l'opportunité d'effectuer un stage de 6 mois sur un chantier de construction et d'encadrer plusieurs fois le cours qu'elle offre à l'École Polytechnique de Montréal, pour la qualité de sa formation et pour la disponibilité de ses experts en ingénierie et planification de projets.

Je remercie aussi tous les étudiants de la Chaire et de Robert Pellerin que j'ai croisé depuis cinq ans pour notre collaboration scientifique, nos échanges intellectuels et leur sympathie : Lucas Grèze, Xavier Guillot, Kaouthar Cherkaoui, Georges Baydoun, Bassem Bouslah, Alban Piveteau, Jean-Mathieu Bieber, Jean-Baptiste Boucherit, Fadwa Houaida, Ariane Duchesne, Andrée-Anne Lemieux, Kenza El Khazraji, Martin Fleuret, Nobel Hardy Takodjou Kengne, Romain Morlhon, Dominique Bernet, Ghassane Lamqari, Kevan Saba, Magno Gaile Gayoso Baez, Thibault Hamon, Cyrill-Alexandre Million.

Je souhaite souligner aussi l'expertise, le soutien et la sympathie de nombreux professeurs et associés de recherche du département de mathématiques et génie industriel de l'École Polytechnique : Gabriel Yan, Carl St-Pierre, Mario Bourgault, Pierre Baptiste, Michel Gamache et Samuel Basseto.

J'adresse également mes remerciements chaleureux aux machines Idra pour leur capacité de calcul en parallèle mises à disposition du CIRRELT et sans qui je n'aurai pas pu mener autant d'expériences scientifiques.

Enfin, je ne peux finir sans remercier ma femme Marie-Élaine et mes deux filles Florence et Charlotte, ma famille et mes amis pour leur encouragement et leur soutien sans faille durant toutes ces années d'étude.

RÉSUMÉ

Le chevauchement d'activités au sein d'un projet est une des techniques les plus répandues pour accélérer l'exécution d'un projet. Le chevauchement d'activités consiste à autoriser des activités qui traditionnellement s'exécutent de façon séquentielle à se chevaucher de sorte que les activités en aval débutent avant la fin des activités en amont en se basant sur des informations partielles. Plusieurs stratégies d'exécution de projets appliquées dans la pratique, telles que l'ingénierie simultanée dans les projets de développement de produits et la construction en « fast-tracking », reposent sur ce principe. Cette technique a montré ses preuves dans sa capacité à réduire la durée de projets, avec cependant plusieurs inconvénients. Le chevauchement peut causer des retouches du travail exécuté à partir d'information préliminaire et mener à des itérations. Ces retouches sont difficilement quantifiables et représentent une charge de travail et des coûts supplémentaires qui peuvent réduire ou annuler les bénéfices du chevauchement. Cela soulève la question de quand et de combien les activités devraient être chevauchées dans les projets industriels. Dans la pratique, les gestionnaires de projets ne possèdent pas d'outils d'aide à la décision pour répondre à cette question.

Cette thèse s'intéresse ainsi au problème d'ordonnancement de projet avec chevauchement d'activités dans un contexte déterministe. Ce problème cherche à déterminer conjointement le meilleur calendrier en termes de coût ou de durée de projet et les décisions de chevauchement, c'est-à-dire quelles activités chevaucher et dans quelle mesure. Nous nous intéressons aux projets complexes caractérisés par des contraintes de disponibilité de ressources, des réseaux complexes d'activités, un nombre important d'activités et de couples d'activités qui peuvent se chevaucher.

Les objectifs de cette thèse sont de modéliser, quantifier et analyser, dans le cas de projets complexes d'ingénierie, l'impact des décisions de chevauchement activités sur les performances de projet (durée et coût), en considérant un modèle réaliste de chevauchement. Cette thèse vise aussi à apporter une meilleure compréhension de ces choix dans les projets complexes et proposer des stratégies générales applicables en pratique.

Les travaux réalisés dans le cadre de cette thèse sont articulés autour de trois articles publiés ou soumis à des revues scientifiques. Le premier article intitulé « *Time-cost trade-offs in resource-constrained project scheduling problems with overlapping modes* » (publié en 2014 dans International Journal of Project Organisation and Management) introduit un modèle de

chevauchement d'activités basé sur des modes de chevauchement reliés aux jalons internes des activités et permet de modéliser de façon réaliste et flexible la relation entre durée de chevauchement et durée de retouche. Ce modèle est inséré dans une modélisation du problème de compromis durée-coût de l'ordonnancement de projets complexes avec contraintes de ressource et chevauchement d'activités sous la forme d'un programme linéaire en nombres entiers. Les durées de communication/coordination et de retouches sont considérées. Le problème est résolu avec une méthode exacte pour un exemple virtuel de projet. Les résultats illustrent les interactions entre le coût de projet, la durée du projet et les contraintes de ressource ainsi que leur influence sur le temps de résolution.

Le second article intitulé « *A Path Relinking-based Scatter Search for the Resource-Constrained Project Scheduling Problem* » (soumis dans European Journal of Operational Research) introduit une métaheuristique dans la famille des recherches dispersées (« *scatter search* ») pour résoudre le problème standard RCPSP (« *Resource-Constrained Project Scheduling Problem* ») sans chevauchement. Cet algorithme utilise la méthode FBI (« *Forward-Backward Improvement* »), inverse la direction d'ordonnancement à chaque itération et est basé sur deux mécanismes novateurs. Premièrement, un PR (« *Path Relinking* ») bidirectionnel avec un nouveau mouvement opérant sur les distances entre activités est utilisé comme méthode de combinaison des solutions. Deuxièmement, une méthode d'amélioration est utilisée pour améliorer la qualité et la diversité des solutions de l'ensemble de référence. Une méthode avancée de paramétrage de l'algorithme utilisant une méthode de recherche locale a été développée pour déterminer les meilleures valeurs de ses paramètres. L'article montre que cette métaheuristique est capable de fournir des solutions de grande qualité avec des temps de calcul acceptables et appartient aux meilleures méthodes approchées existantes dans la littérature pour la résolution des instances virtuelles de projet de PSPLIB.

Enfin, le troisième article intitulé « *Influence of the project characteristics on the efficiency of activity overlapping* » (soumis dans Computers & Operations Research) a pour principales contributions de quantifier et d'analyser l'influence de huit caractéristiques de projets sur l'efficacité du chevauchement pour diminuer la durée de projet. La réduction de la durée de projet est obtenue en résolvant le problème RCPSP avec et sans chevauchement. Deux méthodes de résolution sont développées pour résoudre le problème avec chevauchement. Une nouvelle méthode exacte basée sur un programme linéaire en nombres entiers avec modes de

chevauchement et des techniques de propagation de contraintes est développée. La seconde méthode est une métaheuristique dérivée de la métaheuristique proposée dans le second article. Ces méthodes sont appliquées à un bassin de 3888 instances virtuelles de projets de 30 à 120 activités avec chevauchement. La première observation est que le chevauchement n'apporte aucune réduction dans près de 25% des cas. Une analyse statistique permet de distinguer l'influence de caractéristiques de projets sur l'efficacité du chevauchement et de montrer que la proportion de couples d'activités chevauchables qui sont sur le chemin critique et la sévérité des contraintes de ressources ont le plus d'influence sur la réduction de la durée de projet. Également, les résultats indiquent que certains principes généraux se dégagent pour les décisions de chevauchement. La meilleure stratégie devrait consister à chevaucher peu de couples d'activités chevauchables et de les chevaucher beaucoup. De plus, même si les activités sur le chemin critique sont plus susceptibles d'être chevauchées, les décisions de chevauchement dans un contexte de contraintes de ressource ne doivent pas uniquement être basées sur la criticalité des activités. Enfin, ces observations sont confrontées aux stratégies pratiques de chevauchement proposées dans la littérature.

Ces travaux visent à contribuer au développement d'outils pour assister les gestionnaires de projet dans leurs décisions relatives au chevauchement d'activités. Les principales contributions scientifiques de ces travaux sont les suivantes. Premièrement, nous proposons une modélisation plus réaliste du problème d'ordonnancement de projet avec chevauchement d'activités. Deuxièmement, une nouvelle métaheuristique de type recherche dispersée (« *scatter search* ») performante pour le problème classique RCPSP est développée. Troisièmement, nous introduisons des méthodes de résolution exacte et approchée performantes pour le problème d'ordonnancement de projet avec chevauchement d'activités. La capacité des méthodes exactes à résoudre des problèmes d'ordonnancement pour des projets de grande taille avec contraintes de ressource étant limitée, cette thèse présente en effet à notre connaissance la première méthode approchée de type métaheuristique pour ce problème. Quatrièmement, ces travaux quantifient et analysent l'effet de huit caractéristiques de projet sur l'efficacité du chevauchement d'activités pour diminuer la durée d'un projet. Enfin, nous proposons des principes généraux pour aide les praticiens à prendre les meilleures décisions de chevauchement d'activités.

ABSTRACT

Activity overlapping is one of the most employed strategies used to accelerate project execution. It consists in relaxing the sequential execution of dependent activities by allowing downstream activities to begin before receiving all the final information required from upstream activities. Several practical strategies, such as concurrent engineering and fast-tracking construction, are based on the concept of overlapping. Overlapping has been demonstrated to be powerful for reducing project makespan, but it has some drawbacks. Overlapping often causes additional reworks in downstream activities, as well as iterations of interdependent activities, that are difficult to quantify and represent additional workloads and costs. Such reworks may outweigh the benefices of overlapping in terms of cost and time. This raises the question of when and to which extent overlapping should be applied. In practice, project teams determine overlapping strategies on an ad hoc basis without always considering rework and interaction between activities.

This thesis considers the project scheduling problem with activity overlapping in a deterministic context. This problem aims to jointly determine the best schedule in terms of cost and duration and the best overlapping decisions, namely which activities should be overlapped and to which extent. We focus on the complex projects characterized by constraints on resource availability, a complex network of activities, a large number of activities and a large number of couples of overlappable activities.

The main objectives of this thesis are to model, quantify and analyze the impact of overlapping decisions on the project performances (cost and duration) in the case of complex industrial projects, by considering a realistic model of the overlapping process. This thesis also aims at providing a better understanding of these decisions in complex projects and at guiding planners in improving existing practices.

The research undertaken in this thesis is divided into three papers published or submitted to international peer-reviewed scientific journals. The first paper titled « *Time-cost trade-offs in resource-constrained project scheduling problems with overlapping modes* » (published in 2014 in International Journal of Project Organisation and Management) proposes an overlapping process model based on overlapping modes related to activities' internal milestones that is a

realistic and flexible model of the relation between the amount of overlap and the amount of rework. This overlapping model is then enclosed in a model for the time-cost trade-offs in resource-constrained project scheduling problem with activity overlapping. The model is formulated as a linear integer programming model. The times and costs for communication/coordination and reworks are considered. The problem is solved with an exact method for an illustrative project instance. The results highlight the interactions between the project total cost, its makespan and the severity of the resource constraints and also show their influence on the computational time.

The second paper titled « *A Path Relinking-based Scatter Search for the Resource-Constrained Project Scheduling Problem* » (submitted to European Journal of Operational Research) introduces a metaheuristic based on scatter search for solving the standard RCPSP (Resource-Constrained Project Scheduling Problem) without overlapping. This algorithm involves FBI (Forward-Backward Improvement), reversing the project network at each iteration and two new mechanisms. First, a bidirectional PR (path relinking method) with a new move is used as method for combining solutions. Second, a new improvement procedure is proposed in the reference set update method for enhancing the quality and the diversity of the reference set. An advanced parameter tuning method based on local search is employed. The paper shows that the proposed scatter search produces high-quality solutions in reasonable computational time and is among the best performing heuristic procedures in the literature for solving the instance of the PSPLIB benchmark.

Finally, the main contributions of the third paper titled « *Influence of the project characteristics on the efficiency of activity overlapping* » (submitted to Computers & Operations Research) are to quantify and analyze the influence of eight project characteristics on the efficiency of activity overlapping for reducing project makespan. The reduction of the project makespan is obtained by solving the project scheduling problem with and without overlapping. Two methods have been developed for solving the problem with overlapping. First, we introduce a 0-1 integer linear programming model with overlapping modes and constraint propagation techniques as preprocessing. Second, we propose a metaheuristic based on the scatter search algorithm described in the second paper. These methods are applied on a set of 3888 project instances with overlapping composed of 30 to 120 activities. The first finding is that no reduction of the makespan is observed in about 25% of the projects of the benchmark. A statistical analysis is

conducted to measure the effect of eight project parameters on the makespan gain. It reveals that the proportion of couples of overlappable activities on the critical path and the scarcity of the resource constraints have the highest influence on the makespan gain. In addition, general rules of thumb are derived from the analysis of the results. The best overlapping decisions should consist in overlapping only few couples of overlappable activities and to overlap them with a large degree of overlapping. Even though the activities on the critical path are more likely to be overlapped, overlapping decision should not rely solely on the criticality of the activities. The results are also compared to practical strategies for applying overlapping proposed in the literature.

The main scientific contributions of these works with respect to the scientific literature and from the perspective of assisting project managers to choose the most appropriate overlapping decisions can be summarized as follows. First, we propose a more realistic model of the project scheduling problem with activity overlapping. Second, a new competitive metaheuristic based on scatter search has been developed. Third, we propose competitive exact and heuristic methods for solving the project scheduling problem with activity overlapping. Indeed, as the capacity of exact methods for solving project scheduling problem for large scale projects with resource constraints is limited, the metaheuristic developed in this thesis for this kind of problem is the first in the literature to our knowledge. Fourth, this thesis proposes to quantify and analyze the influence of eight project characteristics on the efficiency of activity overlapping for reducing project makespan. Finally, the findings of this work provide a better understanding of the overlapping decisions and should guide planners for the decisions on activity overlapping.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

ACOSS	Ant Colony Optimization and Scatter Search
AL	Activity List
ALO	Activity List with Overlapping
AOA	Activity On Arc
AON	Activity On Node
ATLAS	Accelerating Two-Layer Anchor Search
B&B	Branch and Bound
CAD	Computer Assisted Design
CAM	Computer Assisted Manufacturing
CAO	Conception Assistée par Ordinateur
CII	Construction Industry Institute
CP	Constraint Propagation
CPM	Critical Path Method
CPU	Central Processing Unit
DABC	Discrete Artificial Bee Colony algorithm
DBGA	Decomposition Based Genetic Algorithm
DSM	Design Structure Matrix
ESS	Enhanced Scatter Search
FAO	Fabrication Assistée par Ordinateur
FBI	Forward-Backward Improvement
FEL	Front-End Loading
FF	Filter-and-Fan approach
GA	Genetic Algorithm

GANS	Genetic Algorithm with Neighborhood Search
GERT	Graphical Evaluation and Review Technique
GPAO	Gestion de la Production Assistée par Ordinateur
IAO	Ingénierie Assistée par Ordinateur
LB	Lower Bound
LPD	Lean Product Development
LFT	Latest Finish Time
LP	Linear Programming
LR	Lagrange Relaxation
LSSPER	Local Search with Subproblem Exact Resolution
NC	Network Complexity
OC	proportion of couples of overlappable activities among all the precedence relations
OL	Overlapping List
MO	Maximum amount of Overlap
MP	Multi-Pass approach
NSERC	Natural Sciences and Engineering Research Council of Canada
PASS	Polarized Adaptive Scheduling Scheme
PBP	Population Based Procedures
PC	proportion of couple of overlappable activities on the critical path of the project without overlapping
PCAP	Parallel Complete Anytime Procedure
PERT	Program evaluation and review technique
PL	Programmation Linéaire
PLM	Product Lifecycle Management
PR	Path Relinking

PLNE	Programmation Linéaire en Nombres Entiers
PPC	Programmation Par Contraintes
PROGEN	Project Scheduling Instance Generator
PSO-HH	Particle Swarm Optimization based Hyper-Heuristic algorithm
PSPLIB	Project Scheduling Problem Library
RBRS	Regret-Biased Random Sampling
RF	Resource Factor
ROC	Receiver Operating Characteristic
RCCP	Rough-Cut Capacity Planning
RCPSP	Resource-Constrained Project Scheduling Problem
RR	Rework Rate
RS	Resource Strength
SA	Simulated Annealing
SAILS	Scatter Search with Adaptive Iterated Local Search
SFLA	Shuffled Frog-leaping Algorithm
SGS	Schedule Generation Scheme
SS	Scatter Search
TO	Topological Order
TOAL	Topological Order Activity List
TS	Tabu Search
UB	Upper Bound
VNS	Variable Neighborhood Search
WBS	Work Breakdown Structure

CHAPITRE 1 INTRODUCTION

Avec l'accroissement de la complexité des projets d'ingénierie, le nombre croissant de projets menés à l'international, combinés avec la volonté de diminuer les coûts indirects des projets et d'assurer un retour sur investissement plus rapide, les compagnies utilisent de plus en plus des stratégies d'accélération de l'exécution des projets (Bogus, Molenaar, & Diekmann, 2005; Terwiesch & Loch, 1999). Une des techniques pour réduire la durée d'un projet consiste à réduire la durée du processus d'ingénierie en appliquant les principes de l'ingénierie simultanée (Lin, Qian, Cui, & Miao, 2010), qui préconise le chevauchement des phases et des activités d'ingénierie. Utilisée dans les projets de développement de produits en industrie, cette stratégie est également commune dans les projets d'ingénierie et de construction dans lesquels elle s'inscrit dans la stratégie plus générale de gestion de projets en régime accéléré (« *fast-tracking* »). Cette stratégie préconise le chevauchement des phases d'ingénierie, d'approvisionnement et de construction (Bogus et al., 2005).

Le chevauchement d'activités consiste à débuter plus tôt des activités à partir d'informations préliminaires provenant d'activités en amont (Krishnan, Eppinger, & Whitney, 1997). Cette pratique permet d'exécuter en parallèle des activités normalement exécutées de manière séquentielle. Les praticiens évaluent le chevauchement comme une pratique largement utilisée qui fait généralement partie de la culture d'une organisation. Cette pratique procure généralement les résultats escomptés en termes de compression de la durée du projet, mais sans garantie sur les coûts et les autres risques. Ces risques peuvent être attribués à la nécessité de reprendre ou retoucher une partie du travail effectué sur la base d'informations incomplètes, ce qui requiert plus de communication et de coordination, et entraîne des heures et des coûts d'ingénierie supplémentaires (Krishnan et al., 1997; Lin et al., 2010). Ces charges de travail supplémentaires peuvent entraîner un retard des activités subséquentes, accaparer des ressources humaines nécessaires à l'exécution d'autres activités du projet, et ainsi entraîner un glissement de l'échéancier du projet. Ceci est valable également à plus haut niveau pour le chevauchement entre les phases de projet. Trop chevaucher n'offre pas nécessairement les meilleures performances de projet en termes de coûts, de durée, et de stabilité dans un contexte incertain (Terwiesch & Loch, 1999).

Négliger ces aspects dans la planification et l'ordonnancement de projet comporte le risque de proposer un échéancier de projet non réaliste qu'il faudra réviser fréquemment durant son exécution. Ces risques sont accentués par le manque d'outils et de méthodes objectives pour guider les praticiens dans le processus de prise de décisions reliées au chevauchement. Dans ce contexte, la quantification et l'analyse de l'impact des décisions de chevauchement sur les performances de projets, et le développement de stratégies de chevauchement effectuant un compromis entre les avantages et inconvénients associés au chevauchement, sont des problèmes cruciaux.

Reconnaissant l'importance des défis associés au chevauchement, ce travail de recherche a pour objectif d'analyser et de quantifier l'impact des décisions de chevauchement entre les activités de projet sur les performances de projet dans le cas de projets complexes d'ingénierie. Notre démarche scientifique et cette thèse s'articule autour de sept chapitres.

Le Chapitre 2 présente les notions de projets complexes d'ingénierie, d'ingénierie simultanée et de construction en « *fast-tracking* » qui constituent le cadre d'étude. Ce chapitre présente aussi une revue de littérature de l'ordonnancement de projet dans un environnement déterministe ou incertain et des modèles de chevauchement existants. Le Chapitre 2 est conclu par une analyse critique de la littérature concernant la prise en compte et l'étude du chevauchement dans l'ordonnancement de projet, en particulier pour les projets complexes.

Le Chapitre 3 présente l'objectif principal et les hypothèses de recherche de cette thèse. Il comporte également la démarche scientifique et la synthèse des travaux menés, et constitue ainsi un fil conducteur cohérent des articles scientifiques. En particulier, ce chapitre présente trois éléments majeurs de nos travaux. Nous présentons en premier lieu les hypothèses de modélisation du chevauchement d'activités au sein d'un projet complexe. En second lieu, nous présentons un modèle innovant de chevauchement d'activités basé sur le concept de modes de chevauchement. Enfin, nous formulons le problème de détermination des décisions de chevauchement et des performances optimales de projet sous la forme d'une extension du problème standard d'ordonnancement de projets avec contrainte de ressources (« *Resource-Constrained Scheduling Problem* », RCPSP).

Les travaux et les contributions scientifiques présentés dans cette thèse s'articulent autour de trois articles scientifiques publiés ou soumis dans des revues scientifiques internationales avec comité

de lecture. Ces trois articles forment le corps de cette thèse dans le Chapitre 4, le Chapitre 5 et le Chapitre 6.

Le premier article intitulé « *Time-cost trade-offs in resource-constrained project scheduling problems with overlapping modes* » (publié en 2014 dans International Journal of Project Organisation and Management) introduit un modèle de chevauchement d'activités basé sur des modes de chevauchement reliés aux jalons internes des activités. Ce modèle est inséré dans une modélisation du problème de compromis durée-coût de l'ordonnancement de projets complexes avec contraintes de ressources et chevauchement d'activités sous la forme d'un programme linéaire en nombres entiers avec modes d'activité. Les durées et coûts de communication/coordination et de retouches sont considérés. Le modèle est résolu avec une méthode exacte pour un exemple virtuel de projet. Les résultats illustrent les interactions entre le coût de projet, la durée du projet et les contraintes de ressource et leur influence sur le temps de résolution.

Le second article intitulé « *A Path Relinking-based Scatter Search for the Resource-Constrained Project Scheduling Problem* » (soumis dans European Journal of Operational Research) introduit une métaheuristique dans la famille des recherches dispersées (« *scatter search* ») pour résoudre le problème standard RCPSP sans chevauchement. Cet algorithme utilise la méthode FBI (« *Forward Backward Improvement* »), inverse la direction d'ordonnancement à chaque itération et est basé sur deux mécanismes novateurs. Premièrement, un PR (« *Path Relinking* ») bidirectionnel avec un nouveau mouvement opérant sur les distances entre activités est utilisé comme méthode de combinaison des solutions. Deuxièmement, une méthode d'amélioration est utilisée pour améliorer la qualité et la diversité des solutions de l'ensemble de référence. Une méthode avancée de paramétrage de l'algorithme utilisant une méthode de recherche locale a été développée pour déterminer les meilleures valeurs de ces paramètres. L'article montre que cette métaheuristique est capable de fournir des solutions de grande qualité avec des temps de calcul acceptables et appartient aux meilleures méthodes approchées existantes dans la littérature pour la résolution des instances virtuelles de projet de PSPLIB.

Enfin, la principale contribution du troisième article intitulé « *Influence of the project characteristics on the efficiency of activity overlapping* » (soumis dans Computers & Operations Research) est de quantifier et d'analyser l'influence de huit caractéristiques de projets sur

l'efficacité du chevauchement pour diminuer la durée de projet. La réduction de la durée de projet est obtenue en résolvant le problème RCPSP avec et sans chevauchement. Deux méthodes de résolution sont développées pour résoudre le problème avec chevauchement. Une nouvelle méthode exacte basée sur un programme linéaire en nombres entiers avec modes de chevauchement et des techniques de propagation de contraintes est développée. La seconde méthode est une métaheuristique dérivée de celle proposée dans le second article. Ces méthodes sont appliquées à un bassin d'instances de projets générées avec chevauchement. Une analyse statistique permet de distinguer l'influence des caractéristiques de projets sur l'efficacité du chevauchement. Également, les résultats indiquent que certains principes généraux se dégagent pour les décisions de chevauchement. Enfin, ces observations sont confrontées aux stratégies pratiques de chevauchement proposées dans la littérature.

Le Chapitre 7 présente une discussion générale de la méthodologie employée et des résultats des travaux de recherche par rapport à la revue critique de la littérature et les problèmes de chevauchement rencontrés dans le milieu industriel. La conclusion dans le Chapitre 8 résume les résultats et la contribution du mémoire et introduit les perspectives de recherches supplémentaires qu'ouvre cette thèse.

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce chapitre présente une revue de la littérature scientifique des travaux abordant l'ordonnancement de projet et le chevauchement d'activités et se structure en 5 parties. Dans une première partie, ce chapitre traite des caractéristiques des projets complexes d'ingénierie en pratique. Dans une seconde partie, ce chapitre résume les travaux scientifiques portant sur la modélisation et la résolution du problème standard d'ordonnancement de projet dans les contextes déterministe et incertain, qui permet d'évaluer la durée des projets. Cette revue de littérature aborde dans une troisième partie les stratégies d'ingénierie simultanée et de construction en « *fast-tracking* », qui constituent le cadre d'application industriel du chevauchement. Une quatrième partie présente une revue des travaux sur la modélisation du chevauchement d'activités et de son intégration dans l'ordonnancement de projets composés de plusieurs couples d'activités chevauchables. Enfin, nous concluons ce chapitre par une analyse critique de la littérature qui nous permet de dégager les opportunités de recherche qui sont traitées dans la suite de cette thèse.

2.1 Projets complexes d'ingénierie

La complexité est maintes fois utilisée pour caractériser les projets, sans que n'en soient fournies de définitions uniformes. Baccarini (1996) propose une définition de la complexité d'un projet comme étant « constituée de plusieurs parties interreliées », qui peut être conceptualisée par les termes « différentiation » et « interdépendance » sur plusieurs dimensions du projet. La différentiation représente par exemple le nombre de niveaux hiérarchiques, d'unités organisationnelles, de spécialités requises, ou le nombre et la diversité des activités du projet. L'interdépendance correspond par exemple au degré d'interaction entre les éléments organisationnels ou entre les activités, notamment dans quelle mesure un changement dans une activité a un impact sur les autres activités.

Pour Williams, T. M. (1999), la différentiation et l'interdépendance ne sont pas suffisantes pour caractériser la complexité d'un projet. Il ajoute une autre composante qui est l'incertitude : incertitude dans les objectifs du projet à atteindre et incertitude dans les méthodes pour atteindre les objectifs. La première reflète notamment dans quelle mesure les spécifications techniques sont correctement définies et figées rapidement. Dans le cas contraire, il y aura beaucoup plus de

demandes de changement durant le projet et donc de retouches et d'itérations, qui sont elles-mêmes d'autant plus nombreuses que les activités sont interdépendantes. Le second type d'incertitude reflète la nouveauté des technologies et le degré de connaissance et d'expertise des intervenants pour répondre aux objectifs, qui, s'il est faible, entraînera une variabilité dans la durée des activités et des erreurs.

De manière plus concrète, on utilisera dans la suite de ce rapport les propriétés suivantes pour caractériser les projets complexes d'ingénierie :

- Taille importante des projets, projets comportant plusieurs centaines ou milliers d'activités,
- Niveau d'incertitude important, du fait des défis techniques et technologiques à surmonter,
- Contraintes de ressources importantes, du fait de la rareté des ressources spécialisées,
- Interactions importantes, entre différentes disciplines d'ingénierie,
- Nombres importants de relations de précédence dans le réseau d'activités,
- Nombre important d'intervenants, internes ou externes à l'organisation,
- Dispersion géographie des intervenants, sur plusieurs sites, parfois dans des pays différents.

Alors que la complexité et l'échelle des projets augmentent, la capacité à compléter avec succès ces projets décroît de manière drastique (Williams, T. M., 1999). La complexité des projets est en outre exacerbée par la tendance actuelle à vouloir réduire la durée des projets, qui oblige à la simultanéité et au parallélisme d'activités qui seraient traditionnellement exécutées en série (chevauchement des activités).

Dans la section suivante est présentée une revue de littérature du problème standard d'ordonnancement de projet avec réseaux complexes d'activités et contraintes de ressource.

2.2 Méthodes classiques d'ordonnancement de projet

Le problème d'ordonnancement de projet avec contraintes de ressources, appelé aussi RCPSP (« *Resource-Consrained Project Scheduling Problem* »), a été abondamment étudié durant ces

dernières décennies. L'ordonnancement consiste à trouver les dates d'exécution des activités et à établir la durée d'un projet en tenant compte des contraintes de ressources et de précédence et en minimisant ou maximisant une fonction objective qui est généralement le coût ou la durée du projet. Cette proposition de recherche étant orientée principalement vers la minimisation de la durée du projet, nous nous restreignons à la présentation des approches qui vise cet objectif. Nous suggérons de consulter Brucker, Drexl, Mohring, Neumann et Pesch (1999) et Hartmann et Briskorn (2010) pour des détails sur la notation et la classification des problèmes d'ordonnancement et les autres variantes.

L'ordonnancement permet de répondre à une des questions majeures auxquelles les gestionnaires et planificateurs de projet doivent répondre : quelle est la durée du projet ? L'établissement de l'échéancier du projet sert aussi de base pour la gestion des ressources humaines, la gestion des ressources matérielles et des équipements, le contrôle et le suivi du projet, l'évaluation de différents scénarios (« *what-if* » scenarios) durant la planification du projet, l'évaluation de l'effet des demandes de changements pendant l'exécution, le règlement des litiges concernant les réclamations pour retard, la prévision et le calcul du cashflow, etc. (Herroelen & Leus, 2005; Kerzner, 2009; Mubarak, 2010).

L'ordonnancement nécessite que les activités (durée, besoin en ressources), les capacités en ressources (humaines ou matérielles) et la séquence logique d'exécution des activités soient définies au préalable. Ces informations sont collectées par les estimateurs et les planificateurs du projet en se basant sur l'historique des projets antérieurs et l'avis d'experts, notamment les futurs exécutants ou responsables des activités. La séquence logique d'exécution est généralement représentée par un réseau AON (« *Activity On Node* ») ou AOA (« *Activity On Arc* »), qui sont des réseaux acycliques dirigés, de sorte que l'exécution du projet s'effectue dans un sens unique sans retour en arrière. Bien que les relations de précédence qui peuvent être modélisées avec ces réseaux sont les relations de dépendance de type fin-début, début-fin, fin-fin, début-début avec ou sans délai, le problème standard du RCPSP se limite aux relations de dépendance de type fin-début sans délai.

Cette section se divise en deux parties. Dans un premier temps sont abordées les méthodes de résolution pour le problème standard RCPSP dans un environnement déterministe. Ensuite sont présentées les approches d'ordonnancement dans un environnement incertain. Alors que le

RCPSP dans un environnement déterministe considère que toutes les données du problème relatives aux activités (durées, besoins en ressources), aux capacités en ressources (humaines ou matérielles) et à la séquence logique d'exécution des activités sont déterministes, l'ordonnancement dans un environnement stochastique considère qu'au moins une de ces composantes est stochastique.

2.2.1 Ordonnancement dans un environnement déterministe

Le problème du RCPSP appartient à la famille des problèmes NP-difficiles au sens fort, il s'agit ainsi d'un des problèmes les plus difficiles à résoudre dans la recherche opérationnelle (Blazewicz, Lenstra, & Rinnooy Kan, 1983). Ceci est dû à la nature cumulative de la consommation des ressources et à l'exécution en parallèle des activités. Ainsi, sans contraintes de ressources, le problème d'ordonnancement peut être aisément résolu à l'aide de l'algorithme du CPM (« *Critical Path Method* ») (Kerzner, 2009). Cette section présente les principales méthodes de résolution exactes ou approchées du RCPSP, ainsi que le RCPSP multi-mode, qui constitue une extension du RCPSP qui peut être appliqué pour réduire la durée d'un projet.

2.2.1.1 Méthodes de résolution exactes

Les méthodes appliquées pour la résolution exacte du RCPSP sont principalement la programmation dynamique (Carruthers & Battersby, 1966; Petrovic, 1968), la programmation linéaire en nombres entiers (PLNE), les méthodes de séparation et évaluation (« *Branch and Bound* », B&B) spécifiques au RCPSP, et la programmation par contraintes (PPC). Notons que la programmation par contrainte (PPC) est en pratique utilisée en combinaison de la PLNE et du B&B. Les méthodes de séparation et évaluation spécifiques et la PLNE sont les approches exactes les plus étudiées dans la littérature sur le RCPSP.

2.2.1.1.1 Méthodes de séparation et évaluation spécifiques au RCPSP

Les méthodes de séparation et évaluation spécifiques au RCPSP consistent à partitionner le problème en sous-problèmes et à éliminer ces sous-problèmes, avec des règles de dominance et en calculant des bornes inférieures. Le processus de séparation en sous-problèmes s'effectue en construisant un arbre de recherche (« *branching* »). Durant l'exécution de l'algorithme, si une borne inférieure d'un sous-problème n'est pas plus petite que la borne supérieure (la meilleure

durée de projet trouvée à date), alors la branche n'est pas explorée plus loin puisque le sous-problème ne peut pas fournir de meilleures solutions. Les différentes méthodes de séparation et évaluation se caractérisent par le schéma de branchement, les règles de dominance et le calcul des bornes inférieures.

Il existe plusieurs approches de branchement qui s'appuient sur les particularités du problème RCPSP: branchements basés sur un arbre de précédence (Patterson, Talbot, Slowinski, & Weglarz, 1990; Sprecher, 2000), branchements basés sur les alternatives avec décalages minimaux (Christofides, Alvarez-Valdes, & Tamarit, 1987; Demeulemeester & Herroelen, 1992), branchements basés sur les alternatives d'échéanciers partiels (Mingozzi, Maniezzo, Ricciardelli, & Bianco, 1998; Stinson, Davis, & Khumawala, 1978) et branchements basés sur les schémas d'ordonnancement (Brucker, Knust, Schoo, & Thiele, 1998).

On distingue deux familles de méthode de calcul de bornes inférieures: les méthodes destructives et constructives. Les méthodes constructives sont obtenues principalement par des relaxations du problème afin d'obtenir un problème moins complexe que le problème original, par exemple la relaxation des contraintes de ressources. D'autres méthodes constructives plus sophistiquées font appel à des approches de génération par colonne et de PPC (Brucker & Knust, 2012). Les méthodes destructives se basent sur le principe que, pour une durée T donnée, s'il n'existe pas de solutions faisables de durées strictement inférieures à T, alors T est une borne inférieure du problème. Il s'agit donc de trouver, à l'aide d'un processus itératif, le plus grand T pour lequel on peut prouver l'infaisabilité, généralement en utilisant la PPC et la programmation linéaire (Brucker & Knust, 2012). Une revue des différentes méthodes constructives et destructives et leur comparaison est présentée dans Klein et Scholl (1999).

Enfin, des règles de dominance sont utilisées afin d'éliminer les branches qui sont dites non-dominantes. En effet, parmi l'ensemble des échéanciers faisables, certains sont presque identiques. Par exemple, pour un échéancier donné, il est parfois possible de décaler vers la gauche certaines activités sans augmenter la durée de l'échéancier. On cherchera donc à éliminer ces redondances dans la recherche d'une solution optimale en se limitant à l'ensemble des échéanciers appelés « actifs » et « semi-actifs », qui contiennent les échéanciers qui ne permettent respectivement aucun déplacement global et local à gauche (« *global and local left shift* »). Les mécanismes de branchement pouvant construire des échéanciers autres qu'actifs ou semi-actifs,

les règles de dominance permettent de restreindre la recherche à ces échéanciers dits dominants. Les règles de dominance utilisées sont les règles « *global left shift rule* », « *local left shift rule* », « *swapping rule* » et « *cutset rule* » (Artigues & Rivreau, 2008; Brucker & Knust, 2012).

2.2.1.1.2 Programmation linéaire en nombres entiers

Les travaux les plus anciens sur la résolution exacte du RCPSP font appel à la PLNE. La principale difficulté réside dans la modélisation des contraintes de ressources au moyen d'inégalités linéaires tout en gardant à tout instant la trace des activités en cours d'exécution. Afin d'y remédier, il est possible soit de discrétiser le temps, entraînant une augmentation du nombre de variables selon l'horizon d'ordonnancement, soit d'utiliser une formulation continue, plus appropriée pour des horizons importants (Artigues et al., 2013; Koné, Artigues, Lopez, & Mongeau, 2011). Parmi les modèles en temps discret, citons la formulation basique de Pritsker, Waiters et Wolfe (1969), la formulation désagrégée (Christofides et al., 1987) et la formulation avec ensembles admissibles de (Mingozzi et al., 1998). Parmi les formulations en temps continu, il existe la formulation avec variables séquentielles (Alvarez-Valdes & Tamarit 1993), la formulation basée sur les flots de ressources (Artigues, Michelon, & Reusser, 2003) et les formulations en temps continu basées sur les évènements (Artigues et al., 2013; Koné et al., 2011).

La programmation linéaire (PL) est facile à résoudre à l'aide de l'algorithme du Simplexe. Lorsque les variables sont entières (PLNE), le problème est difficile à résoudre. Parmi les approches de résolution de la PLNE, les plus performantes font appel à des algorithmes de séparation et évaluation avec typiquement une relaxation continue, en utilisant le Simplexe pour le calcul des bornes inférieures et des méthodes de coupes. L'avantage de la formulation du RCPSP avec la PLNE est la possibilité d'utiliser les méthodes usuelles de résolution des PLNE, qui sont communément implantées dans les solveurs commerciaux tels qu'ILOG-Cplex. Une comparaison de quelques méthodes exactes, effectuée par Koné et al. (2011) et Artigues et al. (2013), montrent d'ailleurs le progrès croissant des performances des solveurs PLNE pour la résolution du RCPSP.

2.2.1.1.3 Limitations, avantages et utilité des méthodes exactes

Herroelen (2005) note que les méthodes exactes sont peu ou pas du tout appliquées dans la pratique de planification. La principale raison réside dans le fait qu'au-delà de 60 activités et avec des contraintes de ressources sévères, ces approches sont incapables de résoudre les problèmes d'ordonnancement (Demeulemeester & Herroelen, 1997). Cependant, outre de garantir l'optimalité, ces méthodes exactes sont intéressantes pour fournir une base de comparaison pour étudier les performances des méthodes approchées, et elles contribuent à une meilleure compréhension du RCPSP en vue de développer des méthodes approchées plus efficaces.

La complexité du RCPSP a ainsi motivé de nombreux chercheurs à privilégier le développement de méthodes approchées de résolution. Parmi ces approches approchées, on distingue principalement les heuristiques basées sur les règles de priorité, aussi appelées méthodes « *X-pass* », et les métaheuristiques (Kolisch & Hartmann, 2006). Notons que la plupart sinon tous les outils de planification commercialisés utilisent des méthodes heuristiques simples basées sur les règles de priorité (Kolisch, 1999).

2.2.1.2 Méthodes heuristiques basées sur les règles de priorité

Les méthodes « *X-Pass* » sont basées sur deux principes : les schémas de génération d'échéancier et les règles de priorité. Un schéma de génération d'échéancier construit un échéancier faisable de manière itérative à partir d'un échéancier partiel, en ajoutant les activités une par une et en déterminant leur date de début. Les deux principaux schémas de génération sont les schémas sériel et parallèle, qui se distinguent par la manière dont l'incrément est fait. Dans le cas du schéma sériel, une unique activité est ajoutée à l'échéancier partiel à chaque itération, alors que, dans le cas du schéma parallèle, un nouveau point de décision dans le temps est considéré à chaque itération pour lequel une ou plusieurs activités sont ajoutées à l'échéancier partiel. La sélection des activités à ajouter est établie par une règle de priorité qui classe les activités selon un critère établi: durée d'exécution la plus petite, marge totale la plus petite, date de début au plus tôt la plus petite, etc. Certaines de ces règles de priorités sont statiques, c'est-à-dire n'évoluant pas au cours de la construction de l'échéancier, alors que d'autres sont dynamiques, ce qui signifie que la liste de priorité des activités est recalculée à chaque itération (Pellerin, 1997). Pour une revue des règles de priorité et pour les détails de l'algorithme des schémas de construction,

nous suggérons au lecteur de consulter Kolisch (1996), Kolisch et Hartmann (1999) et Brucker et Knust (2012).

Les heuristiques basées sur les règles de priorité se caractérisent également par le nombre d'échéanciers générés. On distingue la méthode à passe simple, pour laquelle un seul échéancier est construit avec un schéma de génération et une liste de priorité, et la méthode à passes multiples, qui génèrent plusieurs échéanciers. Pour ces derniers, chaque échéancier est construit à partir de zéro sans considérer aucune information sur les échéanciers générés précédemment. Une possibilité est d'utiliser plusieurs combinaisons de règles de priorités et de schémas de génération différents. Une alternative consiste à incorporer des mécanismes plus élaborés, tels que le biaisement de la règle de priorité (« *sampling methods* ») ou l'alternance de passe avant et passe arrière (« *forward and backward scheduling* »), dans laquelle la passe arrière est l'application du mécanisme de génération d'échéancier au problème miroir obtenu en inversant le réseau d'activités. Ces mécanismes sont détaillés dans Kolisch et Hartmann (1999).

2.2.1.3 Méthodes de résolution approchées - métaheuristiques

De nombreuses métaheuristiques ont été développées pour résoudre le RCPSP, tant dans la famille des méthodes de recherche locale que dans la famille des méthodes de recherche par population. Hartmann et Kolisch (2000) et Kolisch et Hartmann (2006) ont proposé une revue de littérature des métaheuristiques pour le RCPSP. Le premier papier recense 13 méthodes différentes, alors que le second papier en dénombre 37. Depuis ce papier en 2006, nous avons recensé une quarantaine de méthodes additionnelles dans la littérature, ce qui montre l'intérêt exponentiel de la communauté scientifique pour ce sujet. Ces méthodes de plus en plus sophistiquées suivent soit des structures classiques tels que le recuit simulé, la recherche tabou, l'algorithme génétique et la colonie de fourmis, soit combinent des mécanismes issus de différentes structures dans des algorithmes non standards. Nous résumons dans cette section deux méthodes classiques de recherche locale (recuit simulé et recherche tabou), deux méthodes classiques de recherche par population (algorithme génétique et colonie de fourmis) et finissons par les métaheuristiques non standards.

2.2.1.3.1 Métaheuristiques - méthodes de recherche locale

Une des manières les plus courantes d'aborder les problèmes d'optimisation combinatoire est la recherche locale, ou recherche par voisinage (Artigues & Rivreau, 2008; Brucker & Knust, 2012). Il s'agit d'une procédure itérative qui passe d'un échéancier à un autre jusqu'à ce qu'un critère d'arrêt soit satisfait. Appliquée au RCPSP, elle consiste à effectuer des modifications locales à l'aide d'opérateurs de voisinage qui permettent d'intervertir des activités (« *pairwise interchange* », « *swap* », « *shift* »). L'échéancier fournissant la meilleure solution parmi le voisinage est généralement choisi comme nouveau point de départ pour l'itération suivante. Le principal défaut de cette approche est qu'elle ne permet généralement que de trouver un minimum local. Afin de surmonter ce handicap, il est possible d'autoriser des mouvements qui n'améliorent pas la solution, avec le risque que des solutions soient visitées plusieurs fois et qu'un cycle se forme. Les approches de recuit simulé et de recherche tabou sont basées sur de tels mécanismes.

2.2.1.3.1.1 Le recuit simulé

Le recuit simulé est une méthode inspirée de principes de thermodynamique et de processus utilisés en métallurgie qui alternent des cycles de refroidissement lent et de réchauffage tendant à minimiser l'énergie du matériau. L'approche consiste à générer un voisin à partir d'une solution: si ce voisin a une meilleure valeur que la solution actuelle, alors il est accepté; si ce n'est pas le cas, il peut être accepté selon une probabilité qui est strictement décroissante avec E/T. E représente la différence absolue entre les valeurs de la solution actuelle et celle du voisin (niveau d'énergie), et T représente la température actuelle du système, paramètre qui dépend du nombre d'itérations. Au moment où l'algorithme débute, la température est élevée, de sorte que les solutions n'améliorant pas le critère ont une probabilité élevée d'être acceptées. Par la suite, la température baisse de sorte que seules les petites détériorations de la valeur de la fonction-objectif soient acceptées. Introduite par Kirkpatrick, Gelatt et Vecchi (1983), l'approche du recuit simulé a notamment été appliquée au RCPSP par Boctor (1996) et Bouleimen et Lecocq (2003).

2.2.1.3.1.2 La recherche Tabou

La recherche tabou est également une méthode de recherche par voisinage, dans laquelle une mémoire à court terme des récentes modifications locales est utilisée afin d'interdire le retour à

une solution déjà envisagée (liste Tabou). Toutes les solutions du voisinage qui peuvent être générées avec les modifications inscrites dans la liste Tabou ne peuvent être choisies comme nouvelle solution, sauf si elles améliorent la meilleure solution trouvée à date (principe appelé « critère d'aspiration »). La recherche tabou fait également appel à un mécanisme de diversification qui permet de changer de région de recherche si la meilleure solution trouvée à date ne s'améliore pas pendant un certain nombre d'itérations. Pinson, Prins et Rullier (1994) ont été parmi les premiers à utiliser la recherche Tabou pour le RCPSP, qui a ensuite été notamment appliquée par Baar, Brucker et Knust (1998), Artigues et al. (2003) et Kochetov et Stolyar (2003).

2.2.1.3.2 Métaheuristiques - méthodes de recherche par population

Les métaheuristiques basées sur des populations constituent une autre famille de métaheuristiques. Elles englobent les approches évolutionnaires qui font évoluer une population de solutions selon un mécanisme spécifique. À chaque étape du processus, de nouvelles solutions candidates sont générées parmi lesquelles certaines sont sélectionnées pour former la nouvelle population. Nous présentons ici l'algorithme génétique et l'algorithme de colonie de fourmis.

2.2.1.3.2.1 Algorithme génétique

L'algorithme génétique, introduit par Holland (1975) tire son nom du fait qu'elle reproduit les processus d'évolution et de sélection qui caractérise l'évolution biologique. À partir d'une population initiale (première génération), de nouvelles solutions (ou individus) sont produites par croisement entre solutions, et par mutation. Une fois les nouvelles solutions produites, un processus de sélection s'opère pour ramener la population à sa taille initiale en éliminant les solutions les moins « aptes », généralement selon la durée de projet associée. Les algorithmes génétiques ont été appliqués avec succès au RCPSP, par exemple par Hartmann (1998), Hartmann (2002), Alcaraz et Maroto (2001) et Boucherit, Pellerin, Berthaut et Hajji (2010).

2.2.1.3.2.2 Colonie de fourmis

L'optimisation par colonie de fourmis a été développée par Dorigo, Maniezzo et Colorni (1996). Elle reproduit le comportement et les interactions entre des fourmis parcourant des chemins aléatoires à la recherche de nourriture qu'elles marquent avec des phéromones. Celles-ci sont détectées par les autres fourmis, et leur forte concentration les poussera à utiliser un chemin

plutôt qu'un autre. Avec le temps, la concentration en phéromones diminue, tendant à faire disparaître les chemins peu empruntés. La seule application au RCPSP à notre connaissance est celle de Merkle, Middendorf et Schmeck (2002) : à chaque génération, chacune des fourmis construit une solution à l'aide d'un schéma de génération d'échéancier, en sélectionnant les activités selon la combinaison d'une règle de priorité et d'informations sur les phéromones (effet d'apprentissage des précédentes fourmis).

2.2.1.3.3 Métaheuristiques non standards

Les métaheuristiques non standards ne suivent pas des schémas classiques, mais combinent différentes structures de métaheuristiques, des représentations spécifiques et des mécanismes avancées tels que le « *forward-backward improvement* » (FBI) et le « *path relinking* » (PR). En particulier, de plus en plus de méthodes hybrides ont été développées sous la forme d'une métaheuristique de recherche par population dont le mécanisme d'intensification est basé sur une métaheuristique de recherche locale. Ce sont aujourd'hui les méthodes les plus efficaces dans la littérature (Kolisch & Hartmann, 2006). Citons par exemple l'algorithme évolutionnaire de Paraskevopoulos, Tarantilis et Ioannou (2012) qui utilise une nouvelle représentation des solutions sous la forme d'une liste d'évènements (dates de début et de fin des activités), une recherche dispersée (« *scatter search* ») comme structure principale et une méthode de recherche locale comme mécanisme d'intensification.

2.2.1.4 Comparaisons des approches de résolution sur des instances de référence

Une façon de comparer les approches de résolution exactes et approchées est de comparer sur des instances de référence les résultats obtenus en termes de temps de résolution CPU, de mémoire requise et de valeur de la fonction objective. Plusieurs générateurs d'instance ont été développés pour refléter la complexité des projets. Ils sont caractérisés par plusieurs indicateurs exprimant par exemple la complexité des relations de précédence et la sévérité des contraintes de ressources. Bellenguez-Morineau et Néron (2008) résument les caractéristiques de différents générateurs présents dans la littérature. Un des plus populaires est la librairie PSPLIB qui propose des instances de 30 à 120 activités (Kolisch & Sprecher, 1997).

On retrouve des comparaisons et analyses des méthodes exactes dans Herroelen, De Reyck et Demeulemeester (1998), Bellenguez-Morineau et Néron (2008), Koné et al. (2011) et Artigues et

al. (2013), et entre les méthodes approchées dans Kolisch et Hartmann (1999), Hartmann et Kolisch (2000) et Kolisch et Hartmann (2006). Les méthodes exactes sont incapables de trouver les solutions optimales au-delà de 60 activités sur les instances de PSPLIB, soit loin de la taille de nombreux projets réels. Les comparaisons des méthodes heuristiques et métaheuristiques montrent que ces dernières apportent de meilleures solutions, et sont capables de générer des solutions de qualité en quelques secondes, même pour des projets de 300 activités (Debels & Vanhoucke, 2007). De plus, une analyse comparative de 7 logiciels utilisés par les praticiens (MS Project, Primavera, etc.) sur des instances de 30 activités montre que l'écart moyen avec l'optimal est en moyenne de 5.79%, allant de 0% à 51.85% (Kolisch, 1999). Les performances des logiciels décroissent avec le nombre d'activités et la sévérité des contraintes de ressources.

2.2.1.5 Ordonnancement avec activités multi-modes - compression

Parmi les différentes extensions du RCPSP traditionnel, certaines proposent l'exécution des activités selon plusieurs modes, qui correspondent à des combinaisons possibles de durée et ressources, ou de durée et coût requis pour exécuter l'activité. Dans les premier, on parle de RCPSP multi-mode, dans le second cas de compromis durée-coût ou de compression (« *crashing* »). Le problème d'ordonnancement revient alors à déterminer à la fois les dates d'exécution et le mode d'exécution de chaque activité. L'intérêt de ce problème est de pouvoir trouver une durée de projet qui soit plus petite que dans le cas d'un seul mode par activité (RCPSP classique), c'est pourquoi cette thèse aborde cette extension du RCPSP dans la section 2.4.5 sur le chevauchement comme technique d'accélération et dans la section 2.2.2.4 sur les techniques d'ordonnancement applicables lorsque des retards apparaissent suite à des perturbations.

Puisque le problème de compromis durée-coût est considéré généralement sans contraintes de ressources, nous nous concentrons dans ce paragraphe sur le RCPSP multi-mode. La principale différence avec le RCPSP classique est la flexibilité des ressources, qui accroît grandement le nombre d'échéanciers possibles et rend ce problème encore plus difficile à résoudre que le RCPSP classique (Bellenguez-Morineau & Néron, 2008). Les méthodes exactes développées pour aborder ce problème sont principalement des extensions des méthodes spécifiques de séparation et évaluation (Hartmann & Drexl, 1998; Sprecher & Drexl, 1998; Sprecher, Hartmann, & Drexl, 1997). Également, on retrouve dans Herroelen et al. (1998) une formulation de ce

problème avec la PLNE. Les méthodes approchées, que ce soit les heuristiques basées sur des règles de priorité et les métaheuristiques, ont aussi été employées pour le RCPCP multi-mode. Le lecteur est invité à consulter Hartmann et Briskorn (2010) et Kolisch et Padman (2001) pour un état de l'art de ces méthodes.

2.2.2 Ordonnancement dans un environnement stochastique

Durant son exécution, le projet évolue dans un environnement incertain qui vient perturber le bon déroulement du calendrier d'exécution. Les sources de ces incertitudes sont multiples: défis techniques rencontrés, expérience, productivité et disponibilité des ressources humaines, mauvais temps, troubles politiques, retards à la livraison, etc. De plus, l'ordonnancement du projet s'effectue dans les étapes préliminaires du projet alors que la définition du projet et l'ensemble des informations nécessaires pour établir un calendrier d'exécution sont parfois incomplets et approximatifs. Enfin, les demandes de changements durant l'exécution du projet, demandées par le client, la direction, ou les ingénieurs en conception, affectent l'envergure du projet. Certaines activités doivent ainsi être ajoutées et d'autres annulées pour refléter ces changements. En somme, la validité de l'ordonnancement déterministe qui se base sur des données complètes, fiables et un environnement statique et déterministe est remis en cause (Demeulemeester, Herroelen, & Leus, 2008; Goldratt, 1997). Pourtant, la majorité des efforts de recherche a été déployée pour aborder le problème du RCPSP dans un contexte déterministe, et peu de recherche en comparaison a été faite sur l'ordonnancement dans un contexte incertain (Herroelen & Leus, 2005).

Une des méthodes d'ordonnancement en contexte incertain les plus connues est la méthode PERT (« *Program Evolution and Review Technique* »), qui est l'équivalent du CPM pour le contexte incertain (Kerzner, 2009). Elle permet de déterminer la fonction de distribution de la durée du projet et ainsi de calculer par exemple la probabilité de finir le projet dans un délai donné. Mais elle ne définit pas quelles actions prendre pour protéger le calendrier d'un projet face aux incertitudes avant son exécution et/ou réagir durant l'exécution du projet lors de l'apparition des incertitudes. Dans la littérature, plusieurs familles d'approches ont été développées pour répondre à ces problématiques dans un contexte d'incertitude (Herroelen & Leus, 2005). Elles sont résumées dans le Tableau 2.1 et développées dans les sections suivantes.

Tableau 2.1: Approches d'ordonnancement de projet dans un environnement stochastique

Calendrier de base	**Ordonnancement durant l'exécution**
B1) Pas de calendrier de base	E1) Ordonnancement dynamique (politiques d'ordonnancement)
B2) Ordonnancement Prédictif (sans anticipation de la variabilité)	E2) Décisions de gestion
B3) Ordonnancement proactif (robuste)	E3) Ordonnancement réactif

2.2.2.1 Génération d'un calendrier da base

Plusieurs actions sont possibles pour la génération d'un calendrier de base (Demeulemeester et al., 2008). À l'extrême, aucun calendrier de base n'est généré (B1). Une autre approche, l'ordonnancement prédictif, consiste à développer un calendrier de base sans anticipation de la variabilité des paramètres du problème d'ordonnancement (B2), ce qui revient au RCPSP dans un contexte déterministe (section 2.2.1).

Une troisième approche, l'ordonnancement proactif ou robuste (B3) consiste à développer un calendrier de base qui incorpore un certain degré d'anticipation de la variabilité qui aura lieu durant l'exécution du projet (par ex., variabilité de la durée des activités), afin de le protéger autant que possible contre des perturbations. Plusieurs objectifs peuvent être recherchés: la robustesse de la solution (« *solution robustness* »), qui fait référence à l'insensibilité des dates de début des activités, et la robustesse du critère de performance (« *quality robustness* »), qui correspond à l'insensibilité du critère de performances (par ex., la durée du projet). Plusieurs méthodes ont été développées afin de générer un calendrier de projet stable. Elles reposent généralement sur deux étapes (Demeulemeester et al., 2008):

1) Générer un calendrier de base qui ne soit pas protégé contre les incertitudes en utilisant une seule valeur pour la durée de chaque activité (RCPSP dans un contexte déterministe).

2) Améliorer la stabilité du calendrier de base, soit en effectuant une allocation robuste des ressources (Deblaere, Demeulemeester, Herroelen, & Van de Vonder, 2006; Leus &

Herroelen, 2004), c'est-à-dire en déterminant le réseau du flux des ressources qui maximise la stabilité du calendrier, soit en insérant des tampons de temps (« *time buffer* ») dans le calendrier de base (Van de Vonder, Demeulemeester, & Herroelen, 2004, 2005).

Malheureusement, les mesures de robustesse sont très difficiles à évoluer et à minimiser, même pour des problèmes sans contraintes de ressources, étant donné les durées stochastiques des activités (Hagstrom, 1988). C'est pourquoi les techniques de simulation sont souvent utilisées comme étapes intermédiaires. De plus, le problème d'allocation robuste de ressources et celui d'insertion des tampons de temps sont principalement résolus avec des méthodes heuristiques.

2.2.2.2 Ordonnancement durant l'exécution du projet

Une autre distinction est établie par rapport aux décisions prises durant l'exécution du projet pour réagir aux perturbations et choisir quand commencer les activités restantes. L'ordonnancement stochastique ou dynamique (E1) ne génère aucun calendrier initial et utilise une politique d'ordonnancement préétablie pour déterminer en temps réel quelle(s) activité(s) commencer. Une seconde voix consiste à utiliser non plus une politique préétablie, mais plutôt les décisions prises par le gestionnaire de projet durant l'exécution du projet (Décisions de gestion, E2). Une troisième approche, l'ordonnancement réactif (E3) consiste à développer un calendrier de base qui sera révisé ou ré-optimisé durant l'exécution du projet.

2.2.2.2.1 Ordonnancement stochastique/dynamique

L'ordonnancement stochastique/dynamique est un ordonnancement réactif à l'extrême, qui constitue cependant une classe à part en ce qui concerne les modèles et méthodes utilisées. Le problème le plus courant dans la littérature est celui de l'ordonnancement avec durées d'activités incertaines. L'objectif est de minimiser l'espérance mathématique de la durée du projet. Une politique prédéfinie est utilisée durant l'exécution pour choisir quelle(s) activité(s) démarrer parmi les activités faisables (prédécesseurs terminés et ressources disponibles). Ces décisions sont prises en se basant sur les informations disponibles au moment des décisions : le passé, observable, et une connaissance a priori des lois de distribution des durées des activités. Nous référons le lecteur à Herroelen et Leus (2005) pour une revue des modèles et méthodes de résolution reliés à l'ordonnancement stochastique/dynamique.

2.2.2.2.2 Ordonnancement réactif

Les méthodes réactives ne cherchent pas à faire face aux incertitudes durant la création du calendrier de base, mais proposent de le réviser ou le ré-optimiser durant l'exécution du projet lorsqu'un évènement inattendu apparaît. Les méthodes réactives sont capables de traiter un spectre plus large d'évènements incertains que ne le permet l'ordonnancement proactif : addition, soustraction ou modification des contraintes temporelles, des ressources ou des activités (Guéret & Jussien, 2008). Plusieurs questions sont soulevées par cette méthode :

- Quand faut-il produire un nouveau calendrier ? À chaque fois qu'un évènement imprévu se produit, de manière périodique, ou quand un seuil dans la perturbation est atteint.
- Comment le nouveau calendrier est déterminé ? Il s'agit soit de réordonnancer complètement le projet, soit de réparer l'actuel calendrier.

Le réordonnancement complet de projet correspond à un ordonnancement déterministe de la partie du projet qui reste à accomplir à l'instant où la réaction est initiée. Cette approche est rarement employée, puisque le problème du réordonnancement est un problème NP-complet (Picouleau, 1995), et parce qu'il peut apporter des solutions consécutives très différentes (non stables).

L'approche la plus souvent employée consiste donc à réparer le calendrier actuel en essayant de maximiser la stabilité des calendriers de base successifs (nombre de dates de début modifiées, sommes des modifications dans les dates, modifications maximales, nombre de positions relatives des activités qui ont été modifiées entre les calendriers successifs, etc.). Bien que plusieurs méthodes exactes pour ce problème aient été développées, Van de Vonder et al. (2004) ont montré qu'utiliser une méthode exacte de résolution à chaque perturbation devient rapidement infaisable, même pour des petits projets. Ces derniers ont introduit une méthode de résolution heuristique qui utilise, à chaque instant où on décide de réordonnancer, le meilleur calendrier en termes de robustesse de la solution, parmi plusieurs calendriers générés avec des méthodes parallèles et heuristiques utilisant différentes règles de priorité (heuristiques à passes multiples, section 2.2.1.2).

2.2.2.3 Combinaisons d'actions avant et durant l'exécution du projet

Les approches décrites ci-dessous sont combinables afin de proposer des approches intégrées de planification dans un contexte incertain :

- Ordonnancement prédictif-réactif (B2-E3): un calendrier de base prédictif est développé avant le début du projet et peut être révisé ou ré-optimisé durant l'exécution. Notons par exemple la méthode « *Critical chain* » de Goldratt (1997), qui est une des méthodes les plus répandues dans la pratique.
- Ordonnancement proactif-réactif (B3-E3) (Demeulemeester et al., 2008): Cette procédure combine la génération d'un calendrier de base robuste (ordonnancement proactif) à une procédure réactive qui est utilisée lorsque les perturbations ne peuvent être absorbées par le calendrier de base.

2.2.2.4 Autres approches d'ordonnancement

En plus des approches précédemment citées, d'autres approches sont envisageables durant l'exécution du projet afin d'en accélérer l'exécution suite à des retards provoqués par des imprévus: le chevauchement (section 2.4) et la compression d'activités (section 2.2.1.5).

Enfin, alors que l'ensemble de méthodes d'ordonnancement dans un contexte incertain présentées pose comme hypothèse que le réseau logique d'activités est déterministe, le cas des réseaux stochastiques mérite également une attention. Ces réseaux peuvent être représentés par une matrice de décomposition (section 2.4.1) ou par la méthode GERT (« *Graphical Evaluation & Review Technique* »). Cette dernière autorise des branchements probabilistes, conditionnels et des boucles au sein d'un réseau stochastique. Dans l'ordonnancement des réseaux GERT, les activités peuvent être exécutées plusieurs fois, reflétant par exemple les projets de recherche et développement où plusieurs prototypes successifs sont nécessaires pour valider la conception. Une revue de littérature de l'ordonnancement GERT est proposée dans Neumann (1999).

2.3 Stratégies de chevauchement d'activités en pratique : Ingénierie simultanée et construction en fast-tracking

2.3.1 Ingénierie simultanée

Les entreprises évoluent aujourd'hui dans un environnement de concurrence exacerbé et doivent faire face à un marché versatile qui demande des produits de haute qualité rapidement renouvelables. La capacité à réagir rapidement aux attentes changeantes du marché et le délai de mise sur le marché sont devenus des mesures critiques des performances d'une organisation. Cette pression pour obtenir une meilleure qualité et une durée de développement accélérée tout en réduisant les coûts est à l'origine de l'essor de l'ingénierie simultanée depuis les années 1980 pour les projets de développement de produits. L'ingénierie simultanée propose notamment un nouveau modèle de processus de développement basé sur l'interdisciplinarité et le décloisonnement entre les services intervenants durant le projet. Plusieurs définitions du terme « ingénierie simultanée » ou « ingénierie concourante » existent, la plus populaire étant proposée par Winner, Pennell, Bertrand et Slusarczuk (1988):

> "Concurrent engineering is a systematic approach to the integrated, concurrent design of products and their related processes, including manufacture and support. This approach is intended to cause the developers, from the outset, to consider all elements of the product life cycle from concept through disposal, including quality, cost, schedule, and user requirements."

Ceci s'oppose à la vision traditionnelle Taylorienne de l'ingénierie séquentielle qui veut que les différents départements impliqués dans un projet (marketing, bureau d'études, bureau des méthodes, service qualité, services de production, etc.) travaillent de manière séparée et apportent phase par phase leur part de responsabilité dans la vie du produit suivant un ordre logique (Petitdemange, 1991). Dans une telle approche, le cloisonnement des départements et l'impossibilité d'anticiper toute réactivité entraînent de nombreuses itérations qui provoquent un allongement des délais. En effet, ce processus n'est pas complètement séquentiel, car on assiste à des bouclages entre les différents départements. Par exemple, une fois que le produit est testé pour être validé, il faut parfois revoir une partie de la conception. L'ingénierie simultanée propose au contraire de prendre en compte les différents départements, tout au long de ces étapes, en constituant des équipes multidisciplinaires et interfonctionnelles mettant en commun leurs

connaissances pour réfléchir ensemble aux problèmes posés. Ceci permet d'anticiper les retouches, en diminuant les erreurs dès la conception, et de diminuer globalement la durée du cycle de vie du projet, en permettant un chevauchement des phases (Figure 2.1).

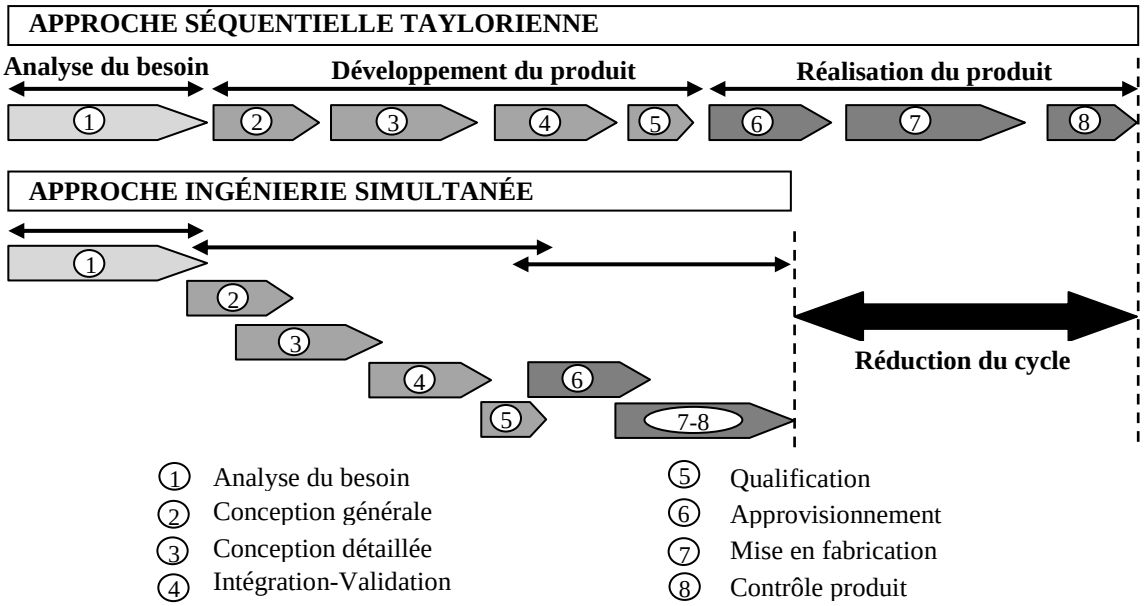

Figure 2.1: Phases du cycle de vie d'un projet de développement de produits

L'ingénierie simultanée comporte plusieurs principes et objectifs clefs, mais elle n'est pas normative concernant la manière de les atteindre (Anumba, Kamara, & Cutting-Decelle, 2007). Néanmoins, un certain nombre d'outils et de méthodes ont été proposés. On retrouve généralement: l'utilisation d'outils de production assistés par ordinateur (CAO, GPAO, etc.), l'application de la conception à coût objectif (« *design to cost* »), la conception en vue de la fabrication (« *design for manufacturing* »), l'ingénierie de la valeur (« *Value engineering* »), le déploiement de la fonction qualité (« *quality function deployment* »), la qualité totale (« *Total Quality Management* ») et l'utilisation de systèmes d'information (Anumba et al., 2007; Bourdichon, 1994; Petitdemange, 1991; Portioli-Staudacher, Van Landeghem, Mappelli, & Redaelli, 2003).

L'ingénierie simultanée exige une coordination étroite entre les différents intervenants du projet. Un des aspects fondamentaux de l'ingénierie simultanée est de faire partager les mêmes données

à l'ensemble des intervenants du projet. Le besoin est de pouvoir disposer d'une information sûre au bon moment, qui circule rapidement, et de connaître la pertinence de cette information, c'est-à-dire son état d'élaboration et de validité. Assurer l'échange d'informations sur tous les axes du projet entre les différents intervenants est l'un des défis rencontrés dans l'ingénierie simultanée (Bourdichon, 1994). La mise à disposition des informations complètes et fiables permet aux décideurs de prendre les bonnes décisions. Ceci est rendu possible par une gestion avisée et rigoureuse de l'information. Les systèmes de gestion de l'information permettent de répondre à ce besoin dans un contexte où le processus de conception s'articule de nos jours autour d'outils informatiques (c.-à-d., conception assistée par ordinateur, CAO). Il est également important, tout au long du projet, de connaître l'état d'évolution des informations qui concoure à l'élaboration progressive du produit. Pour chaque étape du cycle de vie, une référence est constituée qui sert de base informative pour commencer l'étape suivante. La gestion de la configuration garantit que les évolutions de la définition des produits sont identifiées, analysées puis autorisées à être mise en application.

Cependant, bien que ces méthodes et outils soient reconnus comme des facilitateurs pour le succès de l'ingénierie simultanée, de nombreux travaux ont pointé les difficultés à implanter l'ingénierie simultanée dans la pratique (Ford & Sterman, 2003). La réduction du temps de développement visés par l'ingénierie simultanée se fait aux dépens d'un accroissement de la complexité des interactions entre les intervenants et d'une plus grande vulnérabilité aux changements et erreurs, qui requièrent des retouches du fait du chevauchement entre les phases et les activités du projet (Krishnan et al., 1997). En analysant statistiquement les performances de 204 projets de développement de produits dans différentes industries, Terwiesch et Loch (1999) confirment que le chevauchement permet de réduire la durée d'exécution des projets, mais que son efficacité dépend grandement de l'incertitude du projet. Les retouches induites par le chevauchement peuvent annihiler les bénéfices du chevauchement et allonger la durée d'un projet.

2.3.2 Construction en régime accéléré (fast-tracking)

L'application des principes de l'ingénierie simultanée aux projets d'ingénierie et de construction est communément connue sous le terme « *fast-tracking* » (construction en régime accéléré) pour symboliser la réduction de la durée d'un projet de l'ingénierie à la livraison du produit (usine,

bâtiment, etc.). Le processus traditionnel des projets d'ingénierie et de construction se décompose en plusieurs phases exécutées de façon séquentielle: étapes d'avant projet (« *front-end loading* »), ingénierie détaillée, approvisionnement, construction et mise en service. Dans une approche « *fast-tracking* », les phases d'ingénierie détaillée, d'approvisionnement et de construction sont chevauchées (voir la Figure 2.2). Concrètement, la construction commence avant que l'ensemble de l'ingénierie détaillée soit finalisé et validé. Par contre, les phases d'avant-projets sont exécutées de manière séquentielle et font chacun l'objet d'une revue qui autorise ou non la transition vers l'étape suivante (« *stage gate process* »).

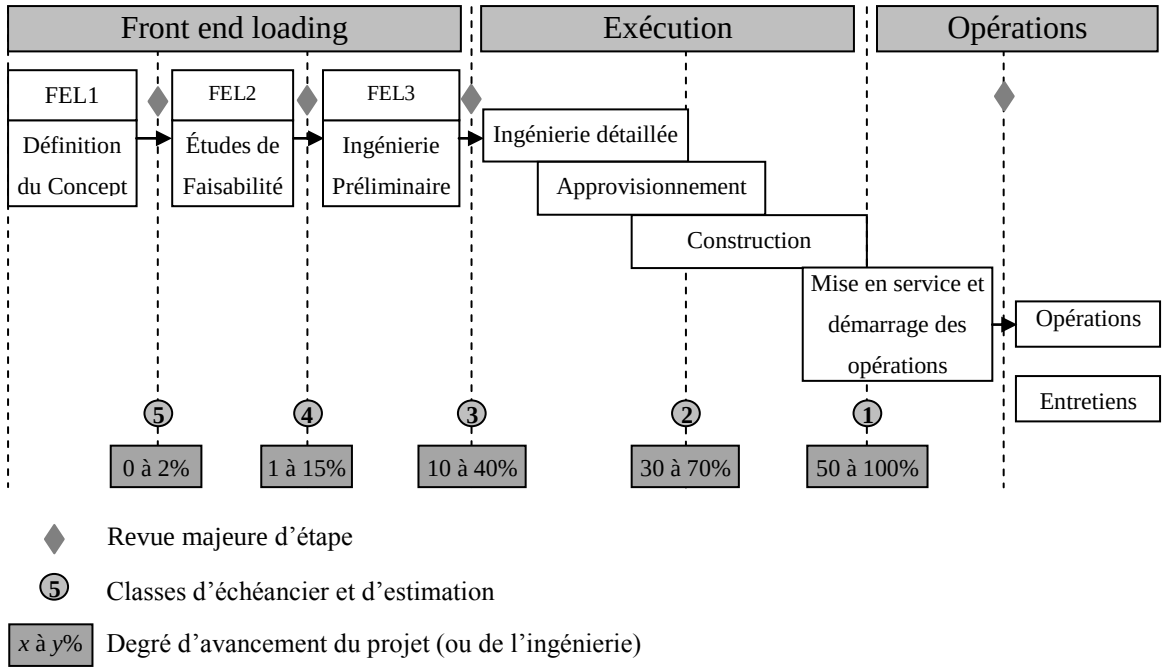

Figure 2.2: Phases d'un projet d'ingénierie et de construction

Bien qu'il n'existe pas de liste reconnue de méthodes et outils spécifiques pour faciliter l'exécution des projets en « *fast-tracking* », plusieurs travaux ont analysé l'influence des bonnes pratiques qui sont généralement reconnues comme efficaces pour les projets traditionnels, dans le cas spécifique des projets en « *fast-tracking* ». Citons notamment Deshpande (2009) qui a proposé une analyse quantitative et qualitative des performances de la phase d'ingénierie détaillée en fonction des bonnes pratiques identifiées dans la littérature ou recommandés par le

CII (Construction Industry Institute), telles que la planification avant-projet (« *front-end loading* »), les relations entre les parties prenantes, les revues de constructibilité, l'alignement, la gestion des changements, la conception en vue de la maintenabilité, la conception en vue de la construction, et la conception en vue de la sécurité. Williams, G. V. (1995) note également le besoin de communication et de confiance accrue entre l'ingénierie et la construction.

Lors de l'exécution en « *fast-tracking* », l'ingénierie détaillée est sous pression dans le but de réduire la durée de conception et permettre à la construction de commencer le plus tôt possible. Une manière de réduire la durée de la phase d'ingénierie est de chevaucher les activités d'ingénierie (Bogus et al., 2005). Dans un contexte de « *fast-tracking* », les activités d'ingénierie occupent ainsi une place accrue, puisque la presque simultanéité entre ingénierie et construction rend la construction plus sensible aux retards de l'ingénierie et aux demandes de changements et retouches en cas d'erreurs (Williams, G. V., 1995). L'interface entre activités d'ingénierie et de construction prend également une place prépondérante dans la construction en « *fast-tracking* », puisque le chevauchement entre les activités d'ingénierie et de construction entraîne le risque de retouches physiques très coûteuses sur les systèmes en cours de construction (Blacud, Bogus, Diekmann, & Molenaar, 2009). La section suivante présente les différents travaux qui proposent des modèles mathématiques visant à déterminer le degré de chevauchement optimal à appliquer lors de la planification d'un projet.

2.4 Modèles de chevauchement d'activités

Le chevauchement d'activités séquentielles apparaît ainsi comme une stratégie fondamentale de l'ingénierie simultanée et de la construction en « *fast-tracking* » pour minimiser la durée d'un projet. Le chevauchement consiste à débuter plus tôt des activités en aval à partir d'informations préliminaires provenant d'activités en amont. Cette stratégie a été décrite comme puissante pour réduire la durée de projets de développement de produits dans plusieurs industries : logiciel (Blackburn, Hoedemaker, & van Wassenhove, 1996), téléphone cellulaire (Lin et al., 2010), automobile (Clark & Fujimoto, 1991) et aéronautique (Sabbagh, 1996). Le chevauchement a aussi été proposé pour les projets de construction (Dzeng, 2006; Pena-Mora & Li, 2001).

Le chevauchement d'activités permet d'exécuter en parallèle des activités normalement exécutées de manière séquentielle (relation fin-début), en échangeant des informations non plus à la date de

fin d'une activité en amont, mais durant toute la durée de chevauchement. Les informations échangées peuvent être tangibles ou intangibles, tels que des plans de conception ou les dimensions de composants, qui constituent les extrants d'une activité en amont et sont requis pour débuter le travail de l'activité en aval. Ainsi, le chevauchement entraîne un accroissement du nombre d'informations échangées et exige une coordination accrue entre les exécutants des activités. Comme l'information fournie par les activités en amont évolue jusqu'à atteindre une forme finalisée, les activités en aval doivent incorporer ces changements au fur et à mesure et se coordonner avec les activités en amont, ce qui entraîne des heures et des coûts d'ingénierie supplémentaires (retouches) et éventuellement de communication et coordination.

La Figure 2.3 représente le processus de chevauchement de deux activités, les retouches et le flux d'information qui en résultent. Si on note par d_i et d_j les durées des activités en amont et en aval, par c_{ij} la durée de chevauchement, et par r_{ij} la durée totale des retouches, alors la durée totale requise pour exécuter les 2 activités peut être exprimé par $d_i + d_j + r_{ij} - c_{ij}$. De même, le gain en temps apporté par le chevauchement peut être exprimé par $r_{ij} - c_{ij}$.

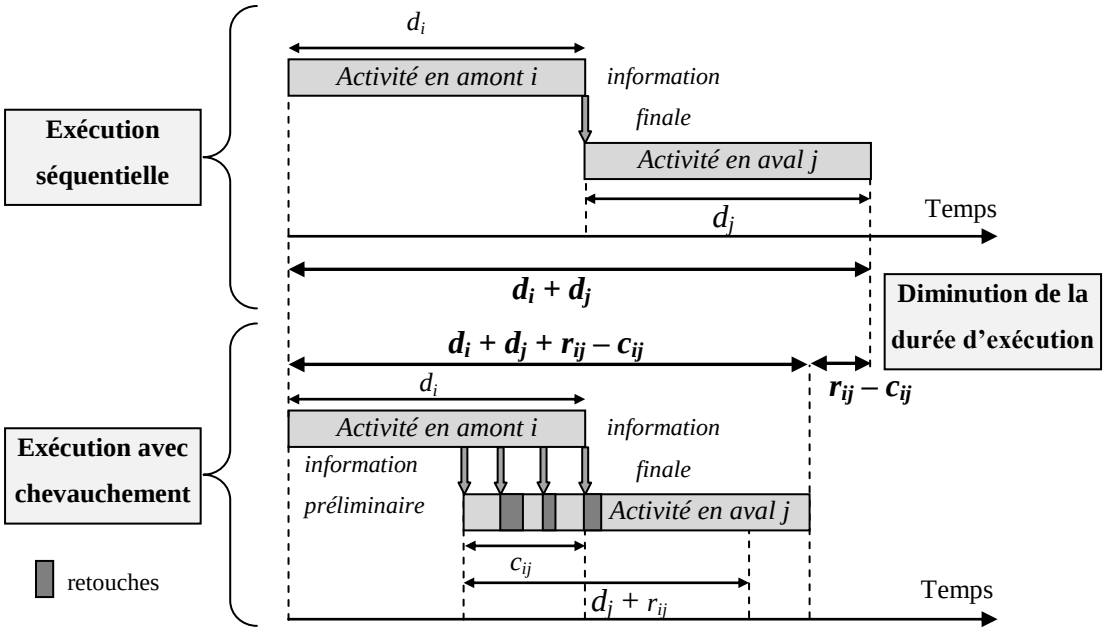

Figure 2.3: Processus de chevauchement de deux activités dépendantes

Cependant, le chevauchement comporte plusieurs inconvénients. Il s'agit d'un processus intrinsèquement risqué puisqu'il suppose de se baser sur des informations préliminaires. Enfin, les retouches causées par le chevauchement peuvent entraîner un retard des activités subséquentes, accaparer des ressources humaines nécessaires à l'exécution d'autres activités, et ainsi entraîner un glissement de l'échéancier du projet.

Il s'agit donc de trouver un compromis dans le chevauchement entre une diminution de la durée du projet et l'augmentation des coûts, de l'incertitude et des retouches: dans quelle mesure faudrait-il chevaucher les activités dans le but de maximiser les performances d'un projet ? Dans la pratique, les praticiens décident du niveau de chevauchement au sein d'un projet sur une base ad hoc. Ils ne possèdent pas de méthodes permettant de quantifier a priori les effets du chevauchement sur les performances d'un projet (Lin et al., 2010).

La littérature scientifique aborde cette problématique sous la forme d'un problème de minimisation de la durée et/ou du coût d'un projet en fonction des décisions de chevauchement d'activités. Les principaux défis de la modélisation du chevauchement dans ce problème sont les suivants :

- Comment modéliser la durée totale des retouches en fonction de la durée de chevauchement pour les activités chevauchées ?
- Comment intégrer les modèles de chevauchement de deux activités à la modélisation d'un projet composé d'un réseau complexe d'activités chevauchables et de contraintes de ressources ?
- Comment peut-on acquérir les données nécessaires aux modèles pour une application industrielle ?

Nous abordons successivement dans les sections suivantes la manière dont la littérature scientifique aborde ces défis.

2.4.1 Représentation des dépendances entre activités à l'échelle d'un projet

La dépendance entre activités peut être caractérisée par l'échange d'information entre elles. Eppinger, Whitney, Smith et Gebala (1994) distinguent trois types de dépendance tels que représentés sur la Figure 2.4 :

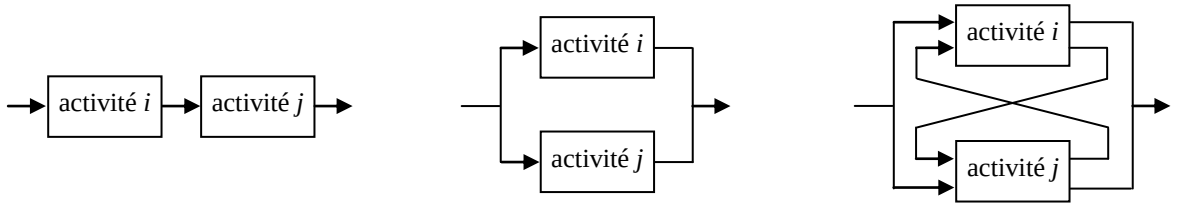

Figure 2.4: Les trois modèles de dépendance entre activités

- les activités dépendantes ou séquentielles, entre lesquels le flux d'information est unidirectionnel;
- les activités indépendantes, qui peuvent être exécutées en parallèle sans interaction;
- les activités interdépendantes, ou couplées, entre lesquels le flux d'information s'effectue dans les deux sens.

Le chevauchement entre activités peut concerner les activités dépendantes et les activités interdépendantes. Les méthodes traditionnelles d'ordonnancement de projets CPM et PERT (voir la section 2.2) sont basées sur des représentations en réseau (AOA, AON) dans lesquels le flux d'information et les relations de précédences sont unidirectionnels. De telles méthodes permettent ainsi de modéliser les activités dépendantes et les activités indépendantes. Les relations entre activités dépendantes peuvent ainsi être représentées par des relations de précédence de type début-début, début-fin, fin-début et fin-fin avec ou sans délai. En particulier, le chevauchement entre activités dépendantes peut être représenté par une relation fin-début avec délais négatifs, de sorte que le délai représente la durée de chevauchement et que la durée de l'activité en amont soit augmentée de la durée de retouches associées.

Pour les activités interdépendantes, des rétroactions d'une activité peuvent entraîner des retouches dans la seconde activité et entraîner ainsi des itérations dans l'exécution de ces activités. Les méthodes d'ordonnancement de projet CPM/PERT ne permettent pas de représenter de telles dépendances et ne peuvent donc pas modéliser les itérations d'activités. Steward (1981) puis Eppinger et al. (1994) ont introduit le concept de matrice de décomposition DSM (« *Design Structure Matrix* ») comme alternative aux représentations en réseau pour inclure la modélisation des activités interdépendantes. Les DSM sont une représentation compacte des dépendances entre

activités sous la forme d'une matrice carrée de dimension égale au nombre d'activités dans un projet. Les éléments DSM_{ij} de cette matrice sont marqués par 1 si de l'information est échangée entre les activités *i* et *j*. Deux activités interdépendantes *i* et *j* sont par exemple modélisés en posant $DSM_{ij} = DSM_{ij} = 1$. Nous référons à Fayez, Axelsson, Oloufa et Hosni (2003) pour une comparaison des représentations CPM/PERT et DSM.

Browning (2001) propose une revue de l'utilisation et de l'application des DSM pour la planification de projet. Il présente la modélisation avec des matrices de décomposition comme une première étape pour représenter les dépendances entre activités, détecter puis minimiser les itérations potentielles. Des techniques de triangularisation des matrices et ultimement l'agrégation ou la décomposition d'activités permettent de minimiser ou éliminer les rétro-informations et donc les itérations (Browning, 2001).

Nous présentons dans la section suivante les modèles d'ordonnancement de projet avec chevauchement d'activités. Ils se distinguent notamment selon la prise en compte ou non des itérations entre activités. Lorsque prises en comptes, les itérations entre activités amènent à une représentation du projet à l'aide de matrices DSM.

2.4.2 Classification des modèles d'ordonnancement de projet avec chevauchement d'activités

Le problème de modéliser et résoudre le problème de minimisation de la durée et/ou du coût d'un projet en fonction des décisions de chevauchement d'activités a été abordé de plusieurs manières dans la littérature scientifique. Il est possible de catégoriser la littérature existante selon les caractéristiques suivantes :

- prise en compte ou non de capacités des ressources disponibles,
- prise en compte ou non des itérations d'activités,
- prise en compte ou non de la durée des retouches,
- modélisation plus ou moins détaillée du chevauchement entre activités,
- réseau d'activités simple ou complexe,

- prise en compte ou non de la durée et/ou du coût des échanges d'information entre activités chevauchées,
- modélisation dans un environnement déterministe ou stochastique.

Certaines de ces caractéristiques sont interreliées. Par exemple, les travaux qui prennent en compte les itérations d'activités les modélisent en utilisant des DSM, utilisent des probabilités d'itération et sont donc des modèles stochastiques. Nous avons donc décidé de regrouper les travaux scientifiques existants en trois catégories qui sont abordés dans les sections suivantes :

- modèles détaillés de chevauchement de deux activités,
- modèles d'ordonnancement déterministe de projets complexes,
- modèles d'ordonnancement stochastique de projets complexes

2.4.3 Modèles détaillés de chevauchement de deux activités/phases

Une première branche de la littérature s'est intéressée au développement de modèles de chevauchement dans le but d'exprimer la quantité de retouches, les coûts additionnels et la durée d'exécution en fonction du degré de chevauchement pour un seul couple d'activités ou phases, ou pour des réseaux de projet simples (activités/phases soit en série soit en parallèle). Ceci permet d'étudier le problème de minimisation de la durée d'exécution et le compromis durée-coût. Ces modèles ont pour avantage de proposer une représentation fine des interactions entre les échanges d'information et l'avancement des activités durant leur exécution. La principale faiblesse de ces modèles est qu'ils ne prennent pas en compte les contraintes de ressources. On peut séparer ces modèles selon qu'ils considèrent les échanges d'information comme étant gratuits et instantanés, ou ayant une durée et un coût non négligeables (par exemple, des réunions). Dans ce dernier cas, le problème d'optimisation est couplé avec la recherche d'une politique de coordination/communication.

Krishnan et al. (1997) ont été des pionniers en proposant un modèle basé sur deux concepts: l'évolution de l'information de l'activité en amont et la sensibilité de l'activité en aval. Le premier caractérise le raffinement de l'information générée et transmise à l'activité en aval, de sa forme préliminaire à sa forme finalisée. Le second représente l'impact d'un changement de cette information sur l'évolution de l'activité en aval (par exemple, la durée pour incorporer ces

modifications). Ils proposent un modèle permettant de déterminer le degré de chevauchement qui minimise la durée d'exécution totale du cycle de développement pour un processus à deux étapes, dans le cas où les échanges d'information sont instantanés. Roemer, Ahmadi et Wang (2000) et Roemer et Ahmadi (2004) ont adapté ce modèle en introduisant le concept de probabilité de retouches en fonction du degré de chevauchement, qui englobe les concepts proposés par Krishnan et al. (1997), pour l'appliquer au problème de compromis durée-coût d'un projet composé de plusieurs phases en série qui peuvent se chevaucher sans itérations. Khoueiry, Srour et Yassine (2013) utilisent également les concepts d'évolution des activités en amont, l'avancement des activités en aval et la sensibilité de ces activités à des changements pour résoudre le problème de maximisation du bénéfice net d'un projet composé de plusieurs activités chevauchables en parallèle.

Lorsque les échanges d'information requièrent des durées et des coûts non négligeables, Loch et Terwiesch (1998) et Lin et al. (2010) proposent de caractériser la politique de communication/coordination par la fréquence et le nombre d'échanges pour le problème de chevauchement et de communication entre deux activités/phases. Loch et Terwiesch (1998) ont adapté les concepts proposés par Krishnan et al. (1997) en considérant l'évolution de l'activité en amont sous la forme d'un taux de modifications. Ils ont analysé le problème de minimisation de la durée en fonction conjointement du chevauchement et de la politique de communication/coordination. Lin et al. (2010) ont raffiné ce modèle en considérant l'évolution de l'avancement de la phase en aval pour le problème de compromis durée-coût. Alors que les travaux ci-dessus proposent des méthodes de résolution exacte, Tyagi, Yang et Verma (2013) introduisent une méthode métaheuristique de type colonie de fourmis pour résoudre le problème de minimisation de la durée à un coût additionnel minimum pour le problème de chevauchement et de détermination de la politique d'échange d'information pour un projet composé de plusieurs activités/phases en série.

La contribution majeure des travaux qui abordent le chevauchement détaillé entre deux activités/phases est de pouvoir modéliser la durée totale des retouches en fonction de la durée de chevauchement. Également, ils concluent que la durée totale des retouches est une fonction croissante et convexe du degré de chevauchement. Si le degré de chevauchement augmente, alors l'activité en aval commence avec des informations moins fiables et plus de changements devront être incorporés. Une autre conclusion importante est que la durée totale d'exécution est une

fonction croissante et convexe du degré de chevauchement si les échanges d'informations sont instantanés et gratuits. Dans le cas où les échanges d'informations ont une durée non négligeable, la durée totale peut être convexe, concave ou convexe-concave selon les caractéristiques de l'évolution de l'information en amont (Loch & Terwiesch, 1998). Ceci signifie concrètement que la politique optimale de chevauchement de deux activités ou phases consiste à chevaucher autant que possible si les échanges d'informations sont instantanés et gratuits, alors qu'il existe un degré de chevauchement optimum qui diffère du chevauchement maximum si les échanges d'informations ont une durée non négligeable.

2.4.4 Modèles d'ordonnancement stochastique de projets complexes avec chevauchement d'activités

Une seconde branche de la littérature regroupe les travaux qui se sont portés sur des projets composés de plusieurs couples d'activités chevauchables au sein d'un projet représenté sous la forme d'un réseau complexe. Ils ont pour point comment de généralement modéliser le chevauchement de couples d'activités de manière plus grossière que dans la section précédente, en prenant l'hypothèse que la durée de retouches est une fonction prédéfinie de la durée de chevauchement et est connue au préalable. À l'aide de technique d'ordonnancement, ces travaux ont étudié l'effet des décisions de chevauchement (quels activités chevaucher et de combien) sur la durée du projet. Les principales faiblesses de ces travaux résident dans le fait qu'ils ne prennent pas en compte les contraintes de ressources, ou que les modèles de chevauchement sont non réalistes (par exemple, la durée totale des retouches est supposée être une fonction linéaire de la durée de chevauchement).

Nous recensons tout d'abord les modèles développés dans un environnement incertain. Ils utilisent généralement des matrices DSM comme support pour représenter les dépendances entre les activités incluant les itérations. Carrascosa (1999) a développé un modèle analytique visant à déterminer la probabilité de finir le projet à l'intérieur d'un délai en fonction des décisions de chevauchement. Ce modèle est cependant limité à des projets de très petite taille (4 à 7 activités) et pour des durées constantes d'activité, étant donné la complexité du problème.

Par la suite, les modèles développés se sont appuyés sur des techniques de simulation dans le but d'approximer la distribution de probabilité de la durée ou du coût du projet, en fonction des

décisions de chevauchement, pour des projets d'une quinzaine d'activités ayant des durées stochastiques. Browning et Eppinger (2002) ont évalué la durée et le coût de projet en utilisant des matrices DSM pour modéliser les probabilités de retouche et d'itérations, ainsi que la proportion du travail dans une activité qui doit être retouchée. La décision de commencer une activité durant l'exécution des simulations est gouvernée par une politique de contrôle qui dépend de l'état du système. Wang, J. et Lin (2009) et Lim, T.-K., Yi, Lee et Arditi (2014) ont raffiné la probabilité de retouches en modélisant explicitement l'évolution des activités en amont. Cho et Eppinger (2005) et Huang et Chen (2006) ont inclus des contraintes de ressources et ont montré que les contraintes de ressource ont un impact sur les décisions de chevauchement et peuvent retarder la date de fin du projet. Nan, He et Han (2013) ont développé un modèle de simulation pour un projet structuré avec une structure de décomposition des travaux (« *Work Breakdown Structure* », WBS) en introduisant une DSM à deux niveaux.

Basés sur la classification des problèmes d'ordonnancement de projet en contexte incertain proposée à la section 2.2.2, ces papiers font partie des problèmes d'ordonnancement dynamique. Ces travaux visent plus à effectuer une analyse de risque basée sur la distribution de probabilité de la durée ou du coût de projet qu'à proposer une action pour faire face aux incertitudes.

2.4.5 Modèles d'ordonnancement déterministe de projets complexes avec chevauchement d'activités

Le problème traditionnel RCPSP déterministe présenté dans la section 2.2.1 a également été étendu pour modéliser le chevauchement d'activités sans considérer ni rétro-information ni itération. Une telle modélisation du problème de chevauchement dans un contexte déterministe vise non seulement à déterminer la durée de projet la plus courte, mais également le calendrier de toutes les activités qui sert de calendrier de référence pour l'exécution du projet et sert de support pour plusieurs fonctions majeures tel que vu à la section 2.2.

Il peut paraître paradoxal d'aborder l'ordonnancement de projet avec chevauchement d'activités dans un contexte déterministe, car le chevauchement est intrinsèquement risqué puisqu'il suppose de se baser sur des informations préliminaires. Cependant, plusieurs approches avancées développées dans la littérature pour l'ordonnancement de projet en contexte incertain, tels que l'ordonnancement proactif et l'ordonnancement réactif, impliquent au préalable la détermination

d'un calendrier de projet sans anticipation des incertitudes (Artigues & Rivreau, 2008; Guéret & Jussien, 2008; Herroelen & Leus, 2004, 2005). La formulation du problème d'ordonnancement de projet avec chevauchement d'activités dans un contexte déterministe constitue ainsi un premier pas vers le développement d'approches d'ordonnancement avec incertitudes.

Bartusch, Mohring et Radermacher (1988) et De Reyck et Herroelen (1998) ont étendu le problème traditionnel RCPSP déterministe en considérant en plus des relations de fin-début sans délai des relations de précédence généralisées. Ces relations de précédence généralisées sont composées des relations de type début-début, début-fin, fin-début, fin-fin avec ou sans délai. Ces relations peuvent donc représenter le chevauchement de deux activités sous la forme d'une relation de précédence de type fin-début avec délais négatifs (correspondant à la durée de chevauchement), mais ces modèles ne considèrent pas d'allongement de la durée de l'activité pour les retouches. On peut donc interpréter ces modèles comme des modèles de chevauchement sans prise en compte des retouches. Bozejko, Hejducki, Uchroski et Wodecki (2014) ont modélisé le problème d'ordonnancement de projet de construction de routes sous la forme d'un problème d'ordonnancement de type flow-shop dans lequel le chevauchement est autorisé. Les contraintes de ressources sont considérées, mais les retouches ne sont pas prises en compte. Le problème est résolu avec une méthode de recherche Tabou.

À notre connaissance, seulement deux travaux ont abordé le problème d'ordonnancement de projet en considérant des retouches non nulles. Plus particulièrement, ces travaux posent comme hypothèse que la relation entre la durée totale de retouches et la durée de chevauchement a été estimée au préalable et est simplement linéaire. En outre, ces travaux proposent d'utiliser en complément d'autres techniques d'accélération de la durée de projets tels que la compression d'activités (« *crashing* », voir la section 2.2.1.5) et la substitution d'activités. Liberatore et Pollack-Johnson (2006) ont ainsi étudié le problème de compromis durée-coût de projet avec chevauchement et compression d'activités sous la forme d'un programme quadratique mixte-entier et sans contraintes de ressources. Seuls Gerk et Qassim (2008) ont étudié le problème avec contraintes de ressources, chevauchement, compression et substitution d'activité, en formulant le problème sous la forme d'un programme linéaire mixte continu basé sur les évènements.

2.5 Stratégies pratiques pour appliquer le chevauchement

En se basant sur les concepts d'évolution de l'information de l'activité en amont et de sensibilité de l'activité en aval à des changements introduits par Krishnan et al. (1997), plusieurs auteurs ont proposé des stratégies pratiques en vue d'améliorer l'efficacité du chevauchement.

Bogus, Molenaar et Diekmann (2006) se sont focalisés sur les stratégies de chevauchement des activités de conception dans les projets d'ingénierie simultanée et de construction en « *fast-tracking* ». Ils ont divisé les stratégies proposées en deux catégories. La première catégorie regroupe les stratégies pour accélérer l'évolution de l'information dans l'activité en amont de sorte de débuter plus tôt l'activité en aval :

- le gel au plus tôt des critères de conception,
- la communication au plus tôt des informations préliminaires de l'activité en amont vers l'activité en aval.
- le prototypage,
- la limitation des itérations et de l'optimisation de la conception au sein de l'activité en amont,
- la standardisation du processus de conception et des composants.

La seconde catégorie de stratégies englobe les stratégies visant à diminuer la sensibilité de l'activité en aval :

- le surdimensionnement de la conception dans l'activité en aval,
- la conception basée sur les ensembles (« *set-based design* »).

Wang, J. et Lin (2009) recommandent les stratégies suivantes pour le développement de nouveaux produits:

- l'utilisation d'outils informatiques avancés telles que la CAO (Conception Assistée par Ordinateur), la FAO (Fabrication Assistée par Ordinateur), la IAO (Ingénierie Assistée par Ordinateur), et l'application des principes du Lean Product Development (LPD) dans le but d'accélérer l'évolution de l'information dans l'activité en amont;

- l'utilisation d'outils de Product Lifecycle Management **(**PLM) et d'ingénierie collaborative pour améliorer la communication entre les équipes de travail des activités en amont et aval et pour limiter les itérations de conception;
- le développement d'architecture de produits modulaires dans lesquels les interactions sont simples et clairement définies, ceci dans le but de diminuer la dépendance entre les étapes de conception et donc la sensibilité de l'activité en aval.

Blacud et al. (2009) se sont intéressés aux stratégies pour améliorer le chevauchement entre les activités d'ingénierie et de construction pour les projets de construction en « *fast-tracking* ». En effet, les risques de retouches peuvent dans ce cas avoir une portée importante, car il s'agit de retouches physiques sur des composants du projet en construction (par exemple, des changements dans la conception de fondations alors que le coulage du béton à débuter). Ils ont identifié quatre facteurs ayant un impact sur la sensibilité des activités de construction en réponse à des changements d'ingénierie: le degré de réversibilité des étapes de construction, le coût et la séquence du processus de transformation au sein de l'activité de construction, la durée d'attente pour remplacer les matériaux et les équipements, la modularité ou le degré d'interaction entre les composants du système en cours de construction, et la nature de ces connexions. À partir de ces facteurs, ils préconisent plusieurs stratégies pratiques pour diminuer la sensibilité des activités en aval :

- la stratégie d'obligation reportée (« *delayed commitment strategy* »), qui consiste à retarder au sein de l'activité de construction en aval les processus de transformation irréversibles,
- la diminution du délai de remplacement des matériaux et des équipements pour les retouches qui requièrent leur remplacement,
- l'amélioration de la modularité du système, de sorte que des changements de conception qui affectent une partie du système en construction ne se répercutent pas sur les autres parties,
- la conception de composants moins sensibles aux changements de conception,
- l'utilisation d'outils de modélisation pour analyser la constructibilité et anticiper les risques de retouches,

- l'amélioration de la communication entre les équipes de conception et de construction, en consultant par exemple les équipes de construction tôt dans le processus d'ingénierie.

Bien que ces auteurs proposent des stratégies concrètes pour améliorer l'efficacité du chevauchement d'activités, l'ordre de grandeur des améliorations possibles et le classement relatif de l'efficacité de ces stratégies ne sont pas discutés.

2.6 Analyse critique de la littérature

Tel que mentionné dans les sections 2.3 et 2.4, si le chevauchement des activités au sein d'un projet permet généralement de minimiser la durée d'un projet, il a pour principaux défauts une augmentation de la complexité et de l'incertitude du projet, du nombre d'informations échangées, et de la quantité de retouches. Peu de chevauchement ne réduit pas suffisamment la durée du projet; trop chevaucher peut entraîner plus de retouches que de gain de temps (Terwiesch & Loch, 1999). La détermination du degré de chevauchement qui permet de faire un compromis et la compréhension entre les interactions des activités sont des problèmes cruciaux, alors que les organisations cherchent en pratique de plus en plus à accélérer la durée d'exécution des projets. Ce constat a mené plusieurs auteurs à développer des modèles mathématiques visant à quantifier l'effet de différents degrés de chevauchement sur les performances d'un projet. Ces travaux présentés à la section 2.4, souffrent cependant de plusieurs lacunes relatives aux points suivants:

La modélisation des durées de retouches en fonction de la durée de chevauchement:

Les modèles peuvent être classés en deux catégories. La première catégorie propose une représentation détaillée des interactions entre les échanges d'information et l'avancement interne des activités en fonction de l'évolution de l'information et de la sensibilité des activités (section 2.4.3), mais ils ne prennent pas en compte les contraintes de ressource et se limitent généralement à des réseaux de projet simples (activités en série par exemple) ou au mieux à des réseaux complexes de projet de moins de 20 activités (Carrascosa, 1999; Lim, T.-K. et al., 2014; Wang, J. & Lin, 2009). L'application de tels modèles détaillés à des projets complexes de grande taille, avec un réseau d'activités complexes et des contraintes de ressources, semble être une voie sans issue.

La seconde catégorie de modèles regroupe les modèles déterministes et stochastiques de projets avec des réseaux complexes d'activités incluant plusieurs couples d'activités chevauchables pour lesquels on pose l'hypothèse que la relation entre degré de chevauchement et quantité de retouches est préalablement connue (sections 2.4.4 et 2.4.5). La principale faiblesse de ces travaux réside dans le fait qu'ils considèrent de façon non réaliste que la durée totale des retouches est supposée être une fonction linéaire de la durée de chevauchement.

Prise en compte des contraintes de ressources :

À l'exception de quelques travaux (Cho & Eppinger, 2005; Gerk & Qassim, 2008; Huang & Chen, 2006), la vaste majorité de la littérature scientifique actuelle ignore les contraintes de ressources. Pourtant Huang et Chen (2006) et Cho et Eppinger (2005) ont montré que les contraintes de ressource ont un impact sur les décisions de chevauchement et peuvent retarder la date de fin du projet. De plus, les contraintes de ressources sont une caractéristique fondamentale des projets complexes en pratique (section 2.1) et forment la principale cause de difficulté à résoudre les problèmes classiques d'ordonnancement de projet (section 2.2). Alors que de nombreux travaux suggèrent d'appliquer le chevauchement spécifiquement aux activités sur le chemin critique (Bogus et al., 2006; Khoueiry et al., 2013; Lim, T.-K. et al., 2014; Wang, J. & Lin, 2009), cette règle ne peut pas s'appliquer dans le cas de contraintes de ressources.

Méthodes de résolution du problème d'ordonnancement déterministe avec chevauchement d'activités:

À notre connaissance, seuls Gerk et Qassim (2008) ont proposé un seul modèle d'ordonnancement déterministe de projet avec chevauchement d'activités qui prennent en compte des durées de retouches et les contraintes de ressources, qui considère toutefois de façon non réaliste que la durée totale des retouches est supposée être une fonction linéaire de la durée de chevauchement. Dans ce papier, le problème de minimiser le coût du projet est résolu avec une méthode de résolution exacte. Cependant, étant donné que le problème d'ordonnancement avec chevauchement d'activités est une extension du RCPSP standard qui est NP-difficile, on peut s'attendre à ce que les méthodes de résolution exactes pour l'ordonnancement avec chevauchement d'activités souffrent des mêmes limitations que le RCPSP standard quant à leur

capacité à résoudre des problèmes de grande taille. Il paraît essentiel de développer des méthodes de résolution approchées efficaces pour le problème d'ordonnancement avec chevauchement d'activités dans le but d'ordonnancer des projets de grande taille tels que rencontrés en pratique.

Modèles d'ordonnancement stochastique avec chevauchement d'activités:

Les travaux présentés à la section 2.4.4 sur les modèles d'ordonnancement stochastique avec chevauchement d'activités visent plus à effectuer une analyse de risque basée sur la distribution de probabilité de la durée ou du coût de projet qu'à proposer une action pour faire face aux incertitudes. À partir de la revue de littérature sur le problème standard d'ordonnancement stochastique de projet présentée à la section 2.2.2, on s'aperçoit pourtant que des approches visant à faire face aux incertitudes existent. En particulier, plusieurs de ces approches se basent sur le développement au préalable d'un calendrier de base déterministe, et utilisent des méthodes approchées dérivées du problème déterministe (par exemple, le choix des tampons de temps pour un ordonnancement proactif). Dans le but de développer dans le futur des approches visant à faire face aux incertitudes dans le problème d'ordonnancement stochastique avec chevauchement d'activités, il apparaît crucial de développer aux préalables des méthodes de résolution efficace pour le problème déterministe.

Stratégies pratiques pour améliorer l'efficacité du chevauchement :

La plupart des travaux d'ordonnancement de projet avec chevauchement d'activités illustrent leur modèle et leur méthode de résolution sur un nombre restreint d'instances de projet. En conséquence, l'analyse des décisions de chevauchement reste spécifique à leurs exemples et il n'est pas possible ni de dériver des enseignements ou des règles généraux sur les meilleures décisions de chevauchement, ni d'analyser l'effet de différentes caractéristiques de projets sur l'efficacité du chevauchement. Nous appelons dans cette thèse à un changement de paradigme afin de proposer des règles générales pour décider du chevauchement d'activités dans des projets avec réseaux complexes d'activités, en se basant sur l'analyse d'un nombre important d'instances de projet. Une telle approche permettrait en outre de contourner le problème de comment acquérir les données nécessaires aux modèles pour une application industrielle, en particulier lorsque le projet à modéliser est nouveau pour l'organisation qui le mène.

2.7 Conclusion

L'étude de l'effet du chevauchement d'activités sur les performances d'un projet (durée, coût) dans le cas de projets complexes d'ingénierie doit, à la lecture de cette revue de littérature, prendre en compte les contraintes de ressources, la multitude des couples d'activités chevauchables, des relations réalistes entre degrés de chevauchement et quantité de retouches, un réseau d'activités complexes, la taille importante des projets complexes. Les limitations présentées ci-dessus dans la littérature scientifique relative au problème d'ordonnancement de projets avec chevauchement constituent des opportunités de recherche qui sont abordées dans la suite de cette thèse. En particulier, la section suivante présente les objectifs de recherche de cette thèse et la méthodologie appliquée pour y répondre.

CHAPITRE 3 MÉTHODOLOGIE DE RECHERCHE

L'objectif de ce chapitre est de présenter les objectifs de recherche de cette thèse suite aux limitations de la littérature scientifique présentée dans le chapitre précédent pour le problème industriel de détermination des décisions de chevauchement dans les projets complexes. Nous abordons ensuite la démarche de recherche pour atteindre ces objectifs de recherche. Les travaux menés dans cette thèse sont présentés sous la forme d'articles séparés dans le Chapitre 4, le Chapitre 5 et le Chapitre 6. Ce chapitre présente aussi une synthèse des travaux qui se veut un fil conducteur des étapes de recherche et des résultats scientifiques d'un article à l'autre.

3.1 Objectifs de recherche

L'objectif principal de cette recherche est d'analyser et de quantifier, dans le cas de projets complexes d'ingénierie, l'impact des décisions de chevauchement entre activités sur les performances de projet (durée et coût) dans un contexte déterministe, en considérant un modèle réaliste de chevauchement. Cet objectif peut être subdivisé en plusieurs sous-objectifs:

- Proposer un modèle réaliste de chevauchement d'activités qui puisse être intégré au problème d'ordonnancement de projet avec chevauchement;
- Développer des méthodes de résolution exactes et approchées pour le problème d'ordonnancement de projet avec chevauchement;
- Quantifier l'impact des décisions de chevauchement d'activités sur l'efficacité du chevauchement en termes de durée de projet en fonction de certaines caractéristiques de projets;
- Analyser les décisions de chevauchement pour apporter une meilleure compréhension de ces choix dans les projets complexes et proposer des stratégies générales applicables en pratique.

Ces objectifs reposent sur les hypothèses scientifiques originales de contribution à la recherche suivantes:

Hypothèse 1 : *Il est possible de modéliser de façon réaliste la durée des retouches en fonction du degré de chevauchement pour les couples d'activités chevauchables.*

Hypothèse 1 : *Il est possible de quantifier l'impact de toutes les décisions de chevauchement sur les performances de projet complexes.*

Hypothèse 3 : *L'efficacité du chevauchement d'activités sur les performances de projet dépend des caractéristiques de projets, notamment de la sévérité des contraintes de ressources.*

Hypothèse 4 : *Il existe des règles générales de chevauchement applicables en pratique qui permettent d'améliorer l'efficacité du chevauchement d'activités.*

3.2 Démarche de recherche

La démarche de recherche de cette thèse pour répondre aux objectifs de recherche se résume par la séquence suivante :

1) Choix des hypothèses réalistes de modélisation du chevauchement d'activités pour les projets complexes;

2) Développement d'un modèle de chevauchement de deux activités qui quantifie de façon réaliste la durée des retouches en fonction du degré de chevauchement;

3) Formulation du problème d'ordonnancement de projets avec chevauchement d'activités avec minimisation de la durée et/ou du coût de projet sous la forme d'un programme linéaire en nombres entiers;

4) Résolution du problème d'ordonnancement de projets avec chevauchement d'activités à l'aide d'une méthode de résolution exacte;

5) Développement d'une métaheuristique dans la famille des algorithmes de recherche dispersée (« *scatter search* ») pour la résolution du problème standard du RCPSP et comparaison avec les autres méthodes de résolution approchées de la littérature scientifique;

6) Adaptation de la métaheuristique pour le problème d'ordonnancement de projets avec chevauchement d'activités;

7) Développement d'un générateur d'instances virtuelles de projet avec chevauchement basé sur plusieurs caractéristiques de projets complexes;

8) Développement d'un bassin complet d'instances virtuelles de projet;

9) Résolution du problème d'ordonnancement de projets avec chevauchement d'activités avec la méthode de résolution exacte et la métaheuristique sur le sous-ensemble du bassin d'instances composé des projets de petite taille, en vue de mesurer l'efficacité de la métaheuristique.

10) Résolution du problème d'ordonnancement de projets avec chevauchement d'activités avec la métaheuristique sur l'ensemble du bassin d'instances de projet;

11) Analyse statistique de l'influence des caractéristiques de projet sur l'efficacité du chevauchement mesurée comme le gain en durée de projet obtenu avec la métaheuristique avec et sans chevauchement;

12) Analyse des décisions de chevauchement et recommandations de stratégies générales de chevauchement;

Les travaux présentés dans cette thèse suivent cette démarche de recherche et sont structurés en trois chapitres. Chaque chapitre est constitué d'un article scientifique publié ou soumis dans des revues scientifiques internationales avec comité de lecture. Le reste de ce chapitre présente une synthèse de ces articles scientifiques qui se veut un fil conducteur des travaux et des contributions de cette thèse.

3.3 Modélisation du problème d'ordonnancement de projet avec chevauchement

Nous présentons dans cette section les principales hypothèses de modélisation, un modèle innovant de chevauchement d'activités basé sur le concept de modes de chevauchement, plusieurs formulations du problème de détermination des décisions de chevauchement et des dates de projet sous la forme d'une extension du problème standard RCPSP.

3.3.1 Hypothèses de modélisation du problème d'ordonnancement de projet avec chevauchement d'activités

Un projet est composé d'un ensemble S composé de n+1 activités incluant deux activités fictives 0 et n+1 qui représentent respectivement le début et la fin du projet. La durée normale d'exécution de chaque activité j est représentée par d_j. Il s'agit de la durée d'exécution lorsque toutes les informations requises des activités précédentes sont considérées comme finalisées et

disponibles au début de l'exécution de l'activité *j* (sans chevauchement et donc sans durée de retouches). Les symboles utilisés dans la modélisation du problème conjoint de minimisation de la durée ou du coût de projet et de détermination des décisions de chevauchement d'activités sont présentés dans le Table 4.1 et le Table 6.1 présentés respectivement dans le Chapitre 4 et le Chapitre 6. En particulier, chaque activité requiert un nombre de ressources dont la disponibilité est limitée. Ce problème est formulé avec les hypothèses suivantes :

1) Le flux d'information entre les activités est unidirectionnel entre les activités en amont et les activités en aval.

Telles que présenté dans la section 2.4 de la revue de littérature, les échanges d'information des activités en aval vers les activités en amont peut mener à des modifications des activités en amont et causer des itérations des activités interdépendantes. Afin d'éliminer ces rétro-informations, il est possible de modéliser les flux d'information du projet sous la forme d'une matrice de décomposition DSM (section 2.4.1) qui sert de base pour déterminer une séquence faisable d'activités sans rétro-informations. Pour cela, il est possible d'appliquer des techniques de triangularisation de matrice et ultimement la décomposition et l'agrégation des activités (Browning, 2001). Nous posons comme hypothèse qu'une telle démarche préliminaire a été conduite et que le projet est constitué uniquement d'activités dépendantes et indépendantes (Figure 2.4).

2) Des informations préliminaires peuvent être échangées entre les activités chevauchables.

Les activités dépendantes peuvent être classées en deux catégories : les activités non-chevauchables et les activités chevauchables. La première catégorie regroupe les activités reliées par une relation de précédence classique de type fin-début sans délai dans laquelle l'activité en aval requière l'information sortante finalisée ou l'achèvement de l'activité en amont. La seconde catégorie inclut les couples d'activités dans lesquelles l'activité en aval peut débuter à partir d'information préliminaire provenant de l'activité en amont en cours d'exécution. Ces couples d'activités chevauchables peuvent être représentées par une relation de précédence de type fin-début avec délai négatif dans lequel le délai correspond à la durée de chevauchement et dépend donc de la décision de chevauchement.

3) Le chevauchement de deux activités peut être modélisé par des modes de chevauchement.

Le modèle de chevauchement de deux activités qui décrit la durée totale de retouches en fonction des durées de chevauchement à l'aide de modes de chevauchement est détaillé à la section suivante.

4) Le problème d'ordonnancement de projets avec chevauchement est formulé dans un contexte déterministe et toutes les données requises sont connues au préalable.

L'ordonnancement de projet est appliqué comme dans la pratique période par période (heure, jour, semaine) (Hartmann, 1999): la disponibilité et l'allocation des ressources sont estimées par périodes, alors que les durées d'activités, les durées de chevauchement et les durées de retouches sont des multiples entiers d'une période. Nous posons comme hypothèse que toutes les données requises pour l'ordonnancement sont connues au préalable, en particulier les durées totales de chevauchement pour chaque mode de chevauchement et pour chaque couple d'activités chevauchables.

Le problème est formulé dans un environnement déterministe et le but est de déterminer non seulement la durée de projet avec chevauchements, mais également les dates de toutes les activités qui forment le calendrier de base et servent de support pour plusieurs fonctions fondamentales de la gestion de projet (section 2.2).

Bien que le chevauchement soit un processus intrinsèquement risqué, puisqu'il suppose de se baser sur des informations préliminaires, la formulation du problème d'ordonnancement de projet avec chevauchement d'activités dans un contexte déterministe constitue un premier pas vers le développement d'approches d'ordonnancement avec incertitudes. En effet, plusieurs approches avancées développées dans la littérature pour l'ordonnancement de projet en contexte incertain et abordées dans la section 2.2.2, tels que l'ordonnancement proactif et l'ordonnancement réactif, impliquent au préalable la détermination d'un calendrier de projet sans anticipation des incertitudes (Artigues & Rivreau, 2008; Guéret & Jussien, 2008; Herroelen & Leus, 2004, 2005).

Les hypothèses ci-dessus sont communes aux deux modèles de chevauchement qui sont présentés dans le Chapitre 4 et le Chapitre 6. Ces deux modèles se distinguent cependant sur la

modélisation des modes de chevauchement, détaillée dans la section suivante, et également dans la durée et le coût des échanges d'informations.

5a) Les échanges d'information entre les activités chevauchés requièrent des durées et des coûts non négligeables (hypothèse applicable au modèle présenté au Chapitre 4).

Alors que les échanges d'information entre des activités qui ne se chevauchent pas s'effectuent uniquement lors de l'achèvement de l'activité en amont, plusieurs échanges d'information préliminaires ont lieu entre les activités chevauchés. Dans le modèle présenté au Chapitre 4, on pose l'hypothèse que ces échanges d'information ont une durée et un coût non-négligeable, correspondant à rencontres interdisciplinaires pour discuter des derniers changements à incorporer dans l'activité en amont (Lin et al., 2010; Loch & Terwiesch, 1998).

5b) Les échanges d'information entre les activités chevauchés requièrent des durées négligeables (hypothèse applicable au modèle présenté au Chapitre 6).

Dans le modèle présenté dans le Chapitre 6, on pose à l'inverse l'hypothèse que les échanges d'information ont des durées négligeables par rapport aux durées d'exécution des activités.

Nous détaillons dans la section suivante un modèle innovant, flexible et réaliste de chevauchement d'activités basé sur le concept de modes de chevauchement.

3.3.2 Modélisation du processus de chevauchement d'activités basé sur le concept de modes de chevauchement

Dans la pratique de la planification et du suivi de projet, l'avancement d'une activité est souvent mesuré par l'accomplissement de jalons internes correspondant à des évènements majeurs : gel des critères de conception, finalisation de l'ingénierie détaillée, finalisation et revue des documents techniques ou tout autres livrables. Ces informations préliminaires sont émises lors des jalons et peuvent être utilisées comme intrants d'une activité en aval. Les retouches du travail accompli par l'activité en aval à partir d'information non finalisée sont définies pour chaque jalon de l'activité en amont. Ce processus est représenté par des modes de chevauchement : il y a

autant de modes de chevauchement que de jalons de l'activité en amont et on associe à chaque mode une durée totale de retouche à réaliser dans l'activité en aval. Ces modes de chevauchement peuvent être également vus comme différentes politiques de chevauchement, depuis une politique de chevauchement agressive à une politique de non-chevauchement.

Les durées totales de retouches associées aux modes de chevauchement ne nécessitent pas d'être linéaires avec les durées de chevauchement, comme considérées de façon simplifiée dans les travaux présents dans la littérature sur l'ordonnancement de projet avec chevauchement (sections 2.4.4 et 2.4.5). En effet, de nombreux travaux sur les modèles détaillés de chevauchement de deux activités ont montré que la durée totale des retouches est une fonction croissante et convexe du degré de chevauchement (section 2.4.3).

Il n'y aucune restriction sur le nombre de successeurs ou de prédécesseurs chevauchés pour une activité. Si une activité en aval est chevauchée par plusieurs activités en amont, la durée totale des retouches est considérée être la somme des durées de retouches causés par chaque activité en amont, tel que considéré dans Cho et Eppinger (2005). Si une activité *j* est à la fois une activité en amont pour un couple (*j*, *l*) et une activité en aval pour un autre couple (*i*, *j*), nous posons comme hypothèse que l'activité *l* de peut commencer avant la fin de l'activité *i* dans le but d'éliminer l'influence du changement d'information de l'activité *i* sur l'activité *l*. Cette hypothèse est connue sous le nom de chevauchement de type sashimi (« *sashimi-style overlapping* ») dans Imai, Ikujiro et Hirotaka (1985) et Roemer et Ahmadi (2004).

Deux modèles de processus de chevauchement différents basés sur les modes de chevauchement sont présentés dans cette thèse dans le Chapitre 4 et le Chapitre 6. Ces modèles sont présentés dans les deux sections suivantes.

3.3.2.1 Modèle de chevauchement d'activités du Chapitre 4

Le premier modèle développé dans cette thèse au Chapitre 4 est représenté à la Figure 3.1 pour le chevauchement de deux activités (*i*, *j*). Un mode de chevauchement est associé à chaque jalon de l'activité en amont *i*, incluant la fin de l'activité. Le nombre total de modes possibles est m_{ij}, incluant le mode sans chevauchement $m = 1$. On note par α_{ijm} la durée de chevauchement, par r_{ijm} la durée totale des retouches et par σ_{ijm} la durée totale de communication/coordination, par Cr_{ijm} le coût total des retouches et par Cc_{ijm} le coût total de communication/coordination associés au

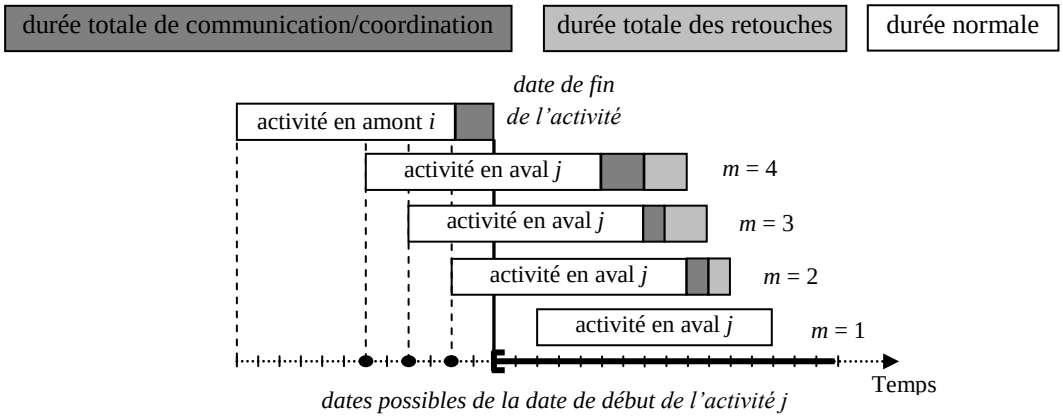

Figure 3.1: Modèle du Chapitre 4 - Processus de chevauchement de deux activités basé sur les modes de chevauchement

mode *m*. Dans le mode $m = 1$, ces paramètres prennent des valeurs nulles ($\alpha_{ij1} = r_{ij1} = \sigma_{ij1} = Cr_{ij1} = Cc_{ij1} = 0$). La durée D_{ijm} et le coût supplémentaire Ct_{ijm} pour exécuter les deux activités (*i, j*), la durée des activités d_{im} et d_{jm} dans le mode *m* peuvent être exprimés de la manière suivante :

$$D_{ijm} = d_i + d_j + r_{ijm} + \sigma_{ijm} - \alpha_{ijm} \quad (1)$$

$$d_{im} = d_i + \sigma_{ijm} \quad (2)$$

$$d_{jm} = d_j + r_{ijm} + \sigma_{ijm} \quad (3)$$

$$Ct_{ijm} = Cr_{ijm} + Cc_{ijm} \quad (4)$$

Les dates possibles de début de l'activité *j* sont exactement les dates des jalons de l'activité en amont si $m > 1$. Si $m = 1$, les dates de début de l'activité *j* et de fin de l'activité *i* sont contraintes par une relation de précédence traditionnelle de type fin-début sans délai.

3.3.2.2 Modèle de chevauchement d'activités du Chapitre 6

Le second modèle développé dans cette thèse au Chapitre 6 est représenté à la Figure 3.2 pour le chevauchement de deux activités (*i, j*). Il y a deux différences par rapport au premier modèle :

- les durées de communication et de coordination entre activités sont considérées comme négligeables dans le modèle du Chapitre 6,
- l'activité en aval *j* peut débuter entre deux jalons de l'activité en amont *i*.

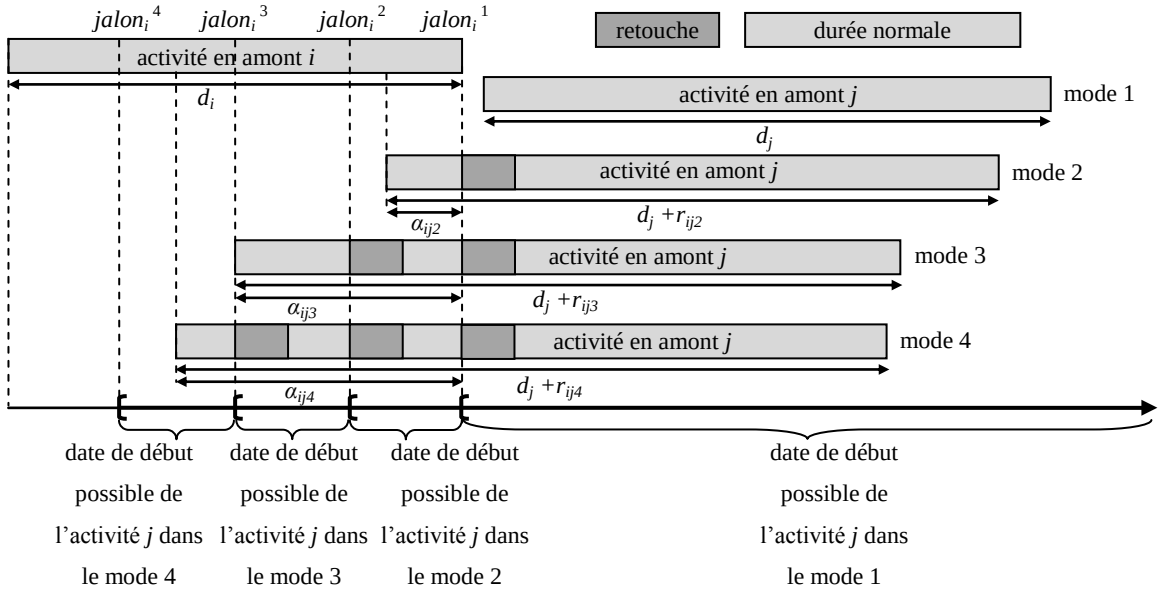

Figure 3.2: Modèle du Chapitre 6 - Processus de chevauchement de deux activités basé sur les modes de chevauchement

Ainsi, si l'activité en aval *j* commence à un jalon de l'activité en amont *i* ou avant le jalon suivant, la même durée totale de retouche est considérée. La durée totale de retouche est donc une fonction constante par morceaux de la durée de chevauchement, dans laquelle chaque marche d'escalier correspond à un mode de chevauchement.

La contrainte entre les dates de jalons de l'activité en amont *i* et la date de début de l'activité *j* dépend aussi du mode de chevauchement. Si elles ne sont pas chevauchées ($m = 1$), les dates de début de l'activité *j* et de fin de l'activité *i* sont contraintes par une relation de précédence traditionnelle de type fin-début sans délai. Si $m \in \{2, ..., m_{ij}\}$, la date de début de l'activité en aval *j* appartient à l'intervalle $[jalon_i^{\,m}, jalon_i^{\,m-1}[$.

3.3.3 Formulations du problème d'ordonnancement de projet avec modes de chevauchement

Trois versions du problème d'ordonnancement de projet avec chevauchement sont présentées dans cette thèse sous la forme de programmes linéaires en nombres entiers, tel que décrit dans le Tableau 3.1. Outre les hypothèses présentées dans les sections précédentes, ces versions se distinguent dans la modélisation et dans la fonction objective.

Tableau 3.1: Différences entre les problèmes d'ordonnancement de projet avec chevauchement abordés dans le Chapitre 4 et le Chapitre 6

<table>
<tr><th></th><th>Modèle 1a</th><th>Modèle 1b</th><th>Modèle 2</th></tr>
<tr><td>Chapitre</td><td colspan="2">Chapitre 4</td><td>Chapitre 6</td></tr>
<tr><td>Hypothèses identiques</td><td colspan="3">1) Le flux d'information entre les activités est unidirectionnel entre les activités en amont et les activités en aval.
2) Des informations préliminaires peuvent être échangées entre les activités chevauchables.
3) Le chevauchement de deux activités peut être modélisé par des modes de chevauchement.
4) Le problème d'ordonnancement de projets avec chevauchement est formulé dans un contexte déterministe et toutes les données requises sont connues au préalable.</td></tr>
<tr><td>Hypothèses différentes</td><td colspan="2">5a) Les échanges d'information entre les activités chevauchés requièrent des durées et des coûts non négligeables.

Une activité en aval j ne peut débuter qu'aux dates jalons de l'activité en amont i.</td><td>5b) Les échanges d'information entre les activités chevauchés requièrent des durées négligeables.

Une activité en aval j peut débuter entre deux jalons de l'activité en amont i.</td></tr>
<tr><td>Modélisation</td><td colspan="2">Modes d'activités</td><td>Modes de chevauchement</td></tr>
<tr><td>Fonction objective</td><td>Maximisation du gain total avec une durée maximale de projet</td><td>Minimisation de la durée de projet avec un gain minimum de projet</td><td>Minimisation de la durée de projet et du chevauchement</td></tr>
</table>

Le Chapitre 4 se concentre sur le problème de compromis durée-coût, ce qui a amené à résoudre le problème avec deux fonctions objectives différentes :

- Maximisation du gain, avec une durée de projet maximale; Le gain est défini par le coût d'opportunité de finir le projet à une date différente (bonus ou malus) et le coût total des retouches et de la communication/coordination pour les activités chevauchées.
- Minimisation de la durée du projet, avec un gain minimum;

Le modèle présenté dans le Chapitre 6 modélise le problème de minimisation de la durée du projet, tout en essayant de chevaucher le moins possible (objectif secondaire en cas d'égalité).

Ces trois problèmes d'ordonnancement de projet avec chevauchement peuvent être exprimés de la façon suivante : *déterminer le calendrier de projet (dates d'exécutions des activités) et les*

décisions de chevauchement (mode de chevauchement) qui optimisent la fonction objective, tout en respectant les contraintes de ressources, de précédence et de chevauchement. Ces trois problèmes sont formulés sous la forme d'un programme linéaire en nombres entiers inspiré du modèle proposé par Pritsker et al. (1969). Les modèles sont présentés dans le Chapitre 4 et le Chapitre 6.

Les deux modèles de programmation linéaire en nombres entiers du Chapitre 4 ont pour particularité de ne pas directement modéliser les modes de chevauchement, mais plutôt des modes d'activités. Les modes d'activités, définis pour chaque activité chevauchable, représente toutes les combinaisons possibles de modes de chevauchement d'une activité avec les activités avec lesquels elle peut chevaucher.

3.4 Résolution du problème d'ordonnancement de projet avec chevauchement

3.4.1 Méthode de résolution exacte

Les trois modèles en programmation linéaire en nombres entiers décrivant les trois versions du problème d'ordonnancement de projet avec chevauchement d'activités ont été implantés dans des logiciels d'optimisation et résolu avec le solveur CPLEX. Plus particulièrement, les deux modèles 1a et 1b du Chapitre 4 ont été implanté dans AMPL v1.6.j et utilisent le solveur CPLEX 12.2. Le modèle 2 du Chapitre 6 a été implanté dans ILOG-CPLEX 12.5 qui utilise le solveur CPLEX 12.5.

Le Chapitre 4 présente une application du problème de compromis durée-coût à un exemple illustratif de 30 activités. Cette application montre que la sévérité des contraintes de ressources et la prise en compte des opportunités de chevauchement ont toutes deux un impact majeur sur les performances du projet en termes de coût et de durée, mais également aussi sur le temps de calcul. Nous obtenons ainsi des temps de calcul supérieurs à 1200 secondes lorsque les contraintes de ressources et le chevauchement sont considérés.

Ces résultats préliminaires confirment nos anticipations que ces deux éléments augmentent la difficulté à résoudre le problème de façon drastique et qu'il faut s'attendre comme pour le RCPSP standard à devoir faire appel à des méthodes de résolution approchée pour des problèmes

de grande taille. Ce constat préliminaire est confirmé dans le Chapitre 6 en résolvant le problème sur un grand nombre d'instances et en appliquant une méthode de résolution exacte plus sophistiquée. Cette méthode de résolution fait appel à la programmation par contraintes décrite dans la section suivante.

3.4.2 Prétraitement à l'aide de la Programmation Par Contraintes

Les modèles en PLNE du Chapitre 4 et du Chapitre 6 utilisent des fenêtres de temps $\{EF_i, ..., LF_i\}$ pour chaque activité i basée sur le CPM pour limiter l'espace de solutions, comme dans la formulation proposée par Pritsker et al. (1969) pour le RCPSP standard. EF_i et LF_i sont respectivement la date de fin au plus tôt et au plus tard de l'activité i. De nombreux travaux pour le RCPSP standard ont cependant proposé d'utiliser la programmation par contraintes (PPC) comme méthode de prétraitement pour dériver des fenêtres de temps plus restreintes, pour détecter l'infaisabilité du problème pour une borne supérieure donnée, et pour renforcer le modèle de PLNE (Artigues et al., 2013; Brucker & Knust, 2000; Brucker & Knust, 2012; Demassey, Artigues, & Michelon, 2005; Koné et al., 2011).

Le principe de la PPC appliqué comme prétraitement peut se résumer comme suit. Le calcul des fenêtres de temps à l'aide de la méthode CPM ne permet pas de prendre en compte les contraintes de ressources dans le RCPSP standard. Pour illustrer comment les contraintes de ressources peuvent être impliquées, considérons deux activités indépendantes (i, j) tels que la somme des ressources requises pour les exécuter dépasse les capacités disponibles. De façon évidente, ces deux activités ne peuvent être exécutées en parallèle. Considérons en plus que les fenêtres de temps calculé avec le CPM sont telles que $LF_i - ES_j < d_i + d_j$, où LF_i et ES_j sont respectivement la date de fin au plus tard de l'activité i et la date de début au plus tôt de l'activité j. Nous pouvons déduire que l'activité i ne peut être exécutée après l'activité j (car la durée totale d'exécution $d_i + d_j$ ne peut avoir lieu dans l'intervalle $\{ES_j, ..., LF_i\}$). Ceci permet de déduire qu'il y a une relation de précédence virtuelle entre les activités i et j. En appliquant de nouveau les passes avant et arrière du CPM avec cette relation additionnelle, les fenêtres de temps peuvent être rétrécis, et de nouvelles relations peuvent être déduites et le processus répété.

Nous avons développé dans le Chapitre 6 un algorithme de PPC pour le problème d'ordonnancement avec chevauchement d'activités. Appliqué au bassin d'instances de projet de 30 activités, nous montrons que l'utilisation de la PPC seule permet de détecter l'infaisabilité de

trouver une durée de projet inférieur à la durée de projet optimal sans chevauchement dans plus de 10% des cas. Utilisé en complément du PLNE, il permet de trouver la solution (calendrier et décisions de chevauchement) optimale en termes de durée de projet dans plus de 97% des instances et de trouver des bornes inférieures de bonne qualité, mais aucune solution faisable pour les instances restantes.

Le temps de calcul moyen est cependant de près de plus de 7000 secondes, avec une distribution des temps de calcul ayant une asymétrie très marquée. Près de 80% des instances sont résolues en moins de 100 secondes, mais près de 3% des instances requièrent un temps de calcul de plus de 100 000 secondes sans forcément obtenir de solutions faisables. Ces résultats obtenus pour des projets de 30 activités montrent les limites des méthodes de résolution exactes pour le problème d'ordonnancement avec chevauchement. Pour ne serait-ce que trouver des solutions faisables pour les cas les plus dures à résoudre avec 30 activités, et pour aborder des projets de plus grande taille, nous avons développé une méthode de résolution approchée décrite dans la section suivante.

3.4.3 Développement d'une métaheuristique de type recherche dispersée

Ces dernières années, de nombreuses métaheuristiques hybrides avancées basées sur les principes de la recherche dispersée (« *scatter search* ») ont été proposées dans la littérature et figurent parmi les méthodes de résolution approchées les plus efficaces (Chen, Shi, Teng, Lan, & Hu, 2010; Debels, De Reyck, Leus, & Vanhoucke, 2006; Mobini, Rabbani, Amalnik, Razmi, & Rahimi-Vahed, 2009; Paraskevopoulos et al., 2012; Ranjbar, De Reyck, & Kianfar, 2009; Valls, Ballestin, & Quintanilla, 2004). Conceptualisée par Glover (1977), la recherche dispersée est une méthode de recherche évolutionnaire qui s'inspire du principe que des méthodes et des mécanismes systématiques pour créer de nouvelles solutions apportent des bénéfices significatifs par rapport à ceux obtenus avec le recours au hasard.

Partant de ce constat, nous avons développé un algorithme de recherche dispersé que nous avons tout d'abord testé sur le RCPSP standard. Les innovations et les performances compétitives de cet algorithme en comparaison de la littérature scientifique sur le RCPSP standard sont présentées dans le Chapitre 5. L'algorithme proposé utilise le FBI comme méthode d'intensification et le PR comme méthode de combinaisons des solutions pour générer de nouvelles solutions (voir la section 2.2.1.3.3). Également, le réseau de projet et la direction

d'ordonnancement sont inversés à chaque itération de l'algorithme. L'algorithme présente deux innovations majeures. Premièrement, nous utilisons un PR bidirectionnel qui utilise un nouveau mouvement. Deuxièmement, une procédure d'amélioration est introduite dans la méthode de mise à jour de l'ensemble de référence. Une méthode de recherche locale englobant la recherche dispersée dans une méta-métaheuristique à deux niveaux est utilisée pour effectuer le paramétrage de l'algorithme de recherche dispersée.

L'algorithme de recherche dispersée adaptée à l'ordonnancement de projet avec chevauchement avec minimisation de la durée de projet et présentée dans le Chapitre 6 utilise les mêmes mécanismes que pour le RCPSP standard, à ceci près que les solutions sont représentées non seulement par une liste d'activité, mais également par une liste des modes de chevauchement représentant les décisions de chevauchement. Le schéma de génération d'échéancier de type sériel, qui permet de calculer un calendrier à partir d'une représentation de solution (voir la section 2.2.1.2), est modifié pour tenir compte des modes de chevauchement. L'application de cet algorithme au bassin d'instances projet de 30 activités et la comparaison des résultats avec la méthode de résolution exacte avec PPC et la PLNE montre que l'écart moyen de la durée de projet obtenu est inférieur à 0.18% avec un temps de calcul bien plus petit de l'ordre de 80 secondes. Dans près de 58% des cas, l'algorithme est capable de trouver la solution optimale en termes de dates d'exécution et de décisions de chevauchement plus rapidement qu'avec la méthode de résolution exacte. En outre, l'algorithme trouve des solutions faisables pour les instances où la méthode de résolution exacte ne pouvait en trouver.

L'algorithme de recherche dispersée appliqué au RCPSP standard et à l'ordonnancement de projet avec chevauchements est donc très efficace puisqu'il permet de trouver des solutions de très bonne qualité avec des temps de calcul acceptable.

3.5 Développement d'un générateur de projets avec chevauchement

Un bassin d'instances virtuelles de projet a été généré dans le but de servir deux fonctions :

- Servir de référence pour l'évaluation de la qualité de la méthode de résolution exacte et de la métaheuristique pour le problème d'ordonnancement de projet avec chevauchement,
- Servir de plan d'expérience pour l'analyse statistique de l'influence de caractéristiques de projet sur l'efficacité du chevauchement (section suivante).

Le générateur a été développé en s'inspirant de PROGEN, le générateur utilisé pour générer la librairie de projets PSPLIB pour le RCPSP standard (Kolisch & Sprecher, 1997; Kolisch, Sprecher, & Drexl, 1995). PSPLIB est basé sur la caractérisation des projets selon quatre paramètres :

- NC (« *Network Complexity* ») : ce paramètre représente le nombre moyen de relations de précédence par activité;
- RF (« *Resource Factor* ») : ce paramètre indique la proportion moyenne de types de ressources différentes requises par une activité parmi tous les types de ressources;
- RS (« *Resource Strength* ») : ce paramètre reflète la sévérité des contraintes de capacité en ressources.
- Le nombre d'activités du projet, de 30 à 120 activités.

Cependant, les instances de PSPLIB ne contiennent aucune donnée relative au chevauchement. Le but du générateur d'instances de projet avec chevauchement est de générer des données de chevauchement à partir d'instances de PSPLIB, selon trois caractéristiques que nous avons définies :

- OC : ce paramètre représente la proportion moyenne de relations de précédence qui peuvent être chevauchées;
- RR : ce paramètre indique la durée de retouche en fonction de la durée de chevauchement. Bien que les méthodes de résolution développée permettent de résoudre des problèmes pour lesquels cette relation n'est pas linéaire, nous considérons dans l'application numérique que cette relation est linéaire avec un taux égal à RR. Ceci permet de caractériser les durées de retouche à l'aide d'un unique paramètre;
- MO : ce paramètre reflète la durée maximum de chevauchement, exprimée en proportion de la durée de l'activité en amont.

Le générateur de projet avec chevauchement est décrit en détail dans le Chapitre 6. Le bassin complet d'instances est créé en générant une instance unique pour chaque combinaison de ces sept caractéristiques de projet. Les combinaisons générées correspondent à un plan d'expérience

complet avec trois ou quatre niveaux par caractéristique. Le bassin complet est composé de 3888 instances virtuelles de projets de 30 à 120 activités.

3.6 Analyse de l'efficacité du chevauchement et développement de stratégies pratiques de chevauchement

Une analyse statistique de l'efficacité du chevauchement en fonction des caractéristiques de projet est présentée dans le Chapitre 6. L'efficacité du chevauchement est mesurée par le gain relatif en durée de projet obtenue en résolvant le problème d'ordonnancement de projet avec et sans chevauchement à l'aide de l'algorithme de recherche dispersée présentée à la section 3.4.3.

Une analyse préliminaire sur le gain observé sur le bassin complet d'instances virtuelles de projet de 30 à 120 activités révèlent que pour près de 25% des instances il n'y a aucun gain avec le chevauchement. Autrement dit, le chevauchement n'est pas efficace pour réduire la durée de projet pour ces projets et chevaucher des activités entraînerait par contre très probablement une augmentation des coûts et plus de communication et coordination. Le gain moyen est de 4.43% avec un écart type de 4.68%. De plus, le gain observé est très dispersé et asymétrique avec des gains allant jusque près de 28%.

Considérant le point masse que constitue les près de 25% de projets ayant un gain nul et l'asymétrie de la distribution du gain observé, une analyse statistique à deux étapes est utilisée (Duan, Newhouse, Morris, & Manning, 1983; Wooldridge, 2002; Yang & Simpson, 2010). La première étape utilise un modèle de régression logistique pour modéliser la probabilité d'observer un gain positif. La seconde étape utilise un modèle linéaire généralisé avec une distribution Gamma et une fonction logarithme comme fonction lien pour modéliser combien est le gain lorsque celui-ci est strictement positif. Ce modèle en deux étapes a un coefficient de régression de près de 75% et montre que toutes les caractéristiques de projets et la plupart des interactions ont une influence sur le gain. En particulier, la proportion de couples chevauchables qui sont sur le chemin critique parmi l'ensemble des couples chevauchables et la sévérité des contraintes de ressources apparaissent comme les paramètres ayant le plus d'influence. Le modèle obtenu peut servir de modèle prédictif pour estimer le gain en durée de projet que l'on peut obtenir en chevauchant, en fonction des caractéristiques de projet.

L'analyse des décisions de chevauchement à l'échelle du bassin complet d'instances virtuelles de projet révèle les enseignements suivants :

- La meilleure stratégie de chevauchement consiste à chevaucher peu de couples d'activité et à les chevaucher beaucoup;
- Les couples d'activités sur le chemin critique ont plus chances d'être chevauchés, mais les décisions de chevauchement ne devraient pas s'appuyer uniquement sur les activités du chemin critique.

Enfin, le Chapitre 6 propose aussi de relier les stratégies pratiques proposées dans la littérature pour améliorer le chevauchement (section 2.5) et les caractéristiques de projet qui ont été analysées. À partir de l'efficacité relative des caractéristiques de projet mesurée avec le modèle a deux étapes, il est possible de dégager quelles stratégies pratiques proposées dans la littérature seraient les plus efficaces.

3.7 Conclusion

Ce chapitre nous a permis de présenter les objectifs de recherche et la démarche de recherche pour atteindre ces objectifs. Nous avons aussi présenté une synthèse des travaux de recherche qui permet de relier les trois articles qui sont présentés dans les trois chapitres suivants.

CHAPITRE 4 ARTICLE 1 : TIME-COST TRADE-OFFS IN RESOURCE-CONSTRAINT PROJECT SCHEDULING PROBLEMS WITH OVERLAPPING MODES

rédigé par:

François Berthaut[1], Robert Pellerin[1], Nathalie Perrier[1] et Adnène Hajji[2]

publié dans :

International Journal of Project Organisation and Management, Vol. 6, No. 3, 2013, p.215-236

DOI: 10.1504/IJPOM.2014.065259

Abstract

In companies, overlapping is commonly regarded as a promising strategy to accelerate project execution. Overlapping consists in executing in parallel two sequential activities by allowing a downstream activity to start before the end of an upstream activity based on preliminary information. However, overlapping can entail reworks in downstream activity, caused by information updates until finalised information is available, and additional coordination and communication, which both require additional time and costs. In this paper, we present a model for the resource-constrained project scheduling problem with feasible overlapping modes. The makespan minimisation and the gain maximisation problems are formulated as linear integer programmes. Time-cost tradeoffs between project duration and overlapping costs are also discussed. An example of a 30-activity project is provided to illustrate the utility and efficiency of the model. Our results highlight the closed interaction between resource constraints and overlapping modes and suggest the relevance of jointly considering them.

[1] Jarislowsky/SNC-Lavalin Research Chair in the Management of International Projects and CIRRELT, Department of Mathematics and Industrial Engineering, École Polytechnique de Montreal, Montreal, Canada (francois.berthaut@polymtl.ca; robert.pellerin@polymtl.ca;nathalie.perrier@polymtl.ca)

[2] Department of Operations and Decision Systems, Faculty of Business Administration, Laval University, Québec, Canada (adnene.hajji@fsa.ulaval.ca)

Keywords: activity overlapping; concurrent engineering; project management; project scheduling; resource constraints.

4.1 Introduction

In recent years, several authors have suggested that the technological products and systems have become more complex, causing an increase of projects in complexity and questioning the relevance of traditional project management approach for dealing with this trend (Baccarini, 1996; Williams, 2002). The importance of understanding complexity in projects has prompted academics to investigate definitions of complexity. Baccarini (1996) defined project complexity as "consisting of many varied interrelated parts" that can be operationalised in terms of differentiation and interdependency in respect to different dimensions, such as organisational complexity (numbers of hierarchical levels, organisational units and specialisations and their interdependencies) and technological complexity (numbers of tasks or specialties and their interdependencies). This definition is also referred as "structural complexity" by Williams (2002) and Remington and Pollack (2007), who defined other types of complexity: 'uncertainty in goals', which refers to unclear or conflicting goals, 'uncertainty in methods' to achieve these goals, induced by design characteristics or technical aspects that are new or unknown, and 'temporal complexity' due to external influences out of the control of a project team. These factors cause perturbations that can trigger feedbacks and reworks because of the number of interdependencies and their natures, particularly the reciprocal interdependencies.

According to Williams (2002), projects have tended to become more time-constrained: time to market is now viewed as a critical factor for success in product development, and firms seek a rapid return on investment (Bogus et al., 2005; Cho & Eppinger, 2005). Several strategies have been developed to achieve an earlier completion date, such as concurrent engineering and fast-tracking. These strategies are based on the concept of overlapping, which consists in starting a downstream activity before receiving all the final information required from upstream activities. Concurrent engineering has been demonstrated to be powerful for reducing product development times from conceptual design to production start-up in a wide range of manufacturing industries (Blackburn et al., 1996; Clark & Fujimoto, 1991; Lin et al., 2010; Sabbagh, 1996). Fast-tracking,

which consists in overlapping design and build phases, is also widely applied in construction projects (Dzeng, 2006; Pena-Mora & Li, 2001).

Overlapping increases the interdependencies and thus the project complexity (Williams, 2002). As preliminary information provided by upstream activities may evolve until it becomes final information, overlapping often causes additional reworks and modifications in downstream activities. Such reworks may outweigh the overlap benefices of parallel activity execution in terms of cost and time (Terwiesch & Loch, 1999). Frequent information exchange between the development teams reduces the negative effect of overlapping, but requires additional communication and coordination. This raises the question of when and to which extent overlapping should be applied. In practice, project teams determine overlapping strategies on an ad hoc basis without always considering rework and interaction between activities, yielding inefficient project management (Lin et al., 2010).

Two groups of models have been developed in the literature to investigate this question. First, many authors consider only a couple of activities or project phases and no resource constraints to establish the best trade-off between overlapping and rework. Krishnan et al. (1997) proposed a model of dependency based on the two concepts of upstream information evolution and downstream sensitivity. Upstream information evolution characterises the refinement of information from its preliminary form to a final value, whereas downstream sensitivity represents the duration of a downstream iteration to incorporate upstream changes. Krishnan et al. (1997) developed a model to determine when to start the downstream activity so as to minimise the development cycle time. Roemer et al. (2000) and Roemer and Ahmadi (2004) adapted this approach to model the probability of rework as a function of the overlap duration and studied the time-cost tradeoffs in overlapping. Loch and Terwiesch (1998) and Lin et al. (2010) investigated the integrated problem of overlapping and information exchange policy, assuming that information exchange usually requires time and cost. In addition to quantifying the amount of rework as a function of the overlap duration, these papers showed that the optimal overlapping strategy is to overlap as much as possible when information exchange is instantaneous and costless, while there exists an optimal overlap duration which differs from the maximum possible overlap duration when information exchange requires non-negligible time and cost.

Other approaches have considered whole projects composed of several couples of overlappable activities, using scheduling techniques to optimise project makespan. De Reyck and Herroelen (1998) and Bartusch et al. (1988) extended the classical resource-constrained project scheduling problem (RCPSP) to consider generalized precedence relations, such as finish-to-start relations with negative time lags, but the proposed models do not consider reworks. Other authors proposed scheduling models under the assumption that the relation between overlapping amount and rework is preliminary known for each couple of overlappable activities. Among them, Liberatore and Pollack-Johnson (2006) proposed a quadratic mixed integer programming model for crashing and overlapping without resource constraints. Gerk and Qassim (2008) developed an analytic project acceleration linear model via activity crashing, overlapping and substitution with resource constraints. Browning and Eppinger (2002), Cho and Eppinger (2005) and Wang and Lin (2009) developed simulation models for the stochastic scheduling problem with overlapping, feedbacks and iterations using design structure matrix (DSM) to represent interdependencies between activities. Resource constraints were only considered by Cho and Eppinger (2005), who showed that these constraints can delay some overlapped activities and delay the project. All these papers assume a simple linear relationship between rework and overlapping amount and consider that information exchange is instantaneous and costless.

In summary, most contributions in the related literature fail to consider a realistic relationship between overlapping, rework and communication/coordination in the RCPSP. The objective of this paper is to extend the classical RCPSP with a realistic overlapping model that deals with additional workloads and costs incurred by rework and coordination/communication. We assume that the information flow is unidirectional from upstream to downstream activities. Consequently, the rework caused by overlapping is only assigned to the downstream activities and there is no activity iteration. The main difference with the aforementioned overlapping models is that overlapping is restricted to a set of feasible overlap durations for each couple of overlappable activities. These overlapping modes are characterised by different overlap durations, rework durations and costs, and communication/coordination durations and costs. For convenience, the overlapping modes are subsequently converted into activity modes, each of which representing a combination of overlapping modes of an activity with the associated overlappable activities. This transformation enables to easily formulate the RCPSP with overlapping modes as a linear integer programming problem.

The remainder of the paper is organised as follows. Section 4.2 first describes the problem statement and assumptions. The gain maximisation and the makespan minimisation problems are formulated in Section 4.3. An illustrative example and computational results of the time-cost trade-offs are then presented in Section 4.4. Section 4.5 concludes with recommendations for future work.

4.2 Problem statement and assumptions

A project is defined by a set of activities, S, including two fictitious activities 0 and $n + 1$, which correspond to the project start and end with zero processing time. We denote by d_j the estimated processing time of activity j when all the final information required from preceding activities are available at its start (i.e., without overlapping). All the symbols and their definitions used along the paper are presented in Table 4.1. We investigate the joint optimisation of overlapping and scheduling with the following assumptions:

- The information flow is unidirectional from upstream to downstream activities.

Feedback information exchange from downstream activities leads to modifications performed by the upstream activities, and iterations can virtually occur (Wang & Lin, 2009). In order to eliminate feedbacks, DSM can be used to represent the relations between activities and to determine a feasible sequence of activities without any feedback, using block triangularisation algorithm (Browning, 2001). As a last resort, activities can be aggregated or decomposed to eliminate them. We assume that such preliminary studies have been conducted. The project is then only composed of independent and dependent activities.

- Preliminary information can be exchanged between identified overlappable activities.

The dependent couple activities can be categorised into non-overlappable and overlappable ones. The former represents the case where a downstream activity requires the final output information from an upstream activity or its completion. These activities are connected with the classical finish-to-start precedence constraint. The latter represents the case where a downstream activity can begin with preliminary information before an upstream activity is finished. The overlappable activities are connected with a finish-to-start-plus-lead time precedence constraint where the lead-time accounts for the amount of overlap.

Table 4.1: Symbols and definitions

Symbol	Definition
S	Set of activities
n	Number of non-dummy activities
E	Set of temporal or precedence constraints
$i \rightarrow j$	Precedence constraint
d_j	Processing time of activity j
A	Set of couples of overlappable activities
P	Set of couples of non-overlappable activities
Po_j	Set of immediate predecessors of activity j that are overlappable with activity j
Pn_j	Set of immediate predecessors of activity j that are not overlappable with activity j
P_j	Set of immediate predecessors of activity j
So_j	Set of immediate successors of activity j that are overlappable with activity j
Sn_j	Set of immediate successors of activity j that are not overlappable with activity j
S_j	Set of immediate successors of activity j
R	Set of renewable resources
R_k	Constant amount of available units of renewable resource k
R_{jk}	Per period usage of activity j of resource k
m_{ij}	Number of precedence modes of the couple (i, j)
α_{ij}	Overlap duration between activities i and j
r_{ij}	Rework in the downstream activity j when (i, j) are overlapped
σ_{ij}	Amount of time for coordination and communication when (i, j) are overlapped
α_{ijm}	Overlap duration between activities i and j in precedence mode m, expressed as a fraction of d_j
r_{ijm}	Rework in the downstream activity j when (i, j) are overlapped in precedence mode m
σ_{ijm}	Amount of time for coordination and communication when (i, j) are overlapped in precedence mode m
Cr_{ijm}	Rework cost in the downstream activity j when (i, j) are overlapped in precedence mode m
Cc_{ijm}	Cost for coordination and communication when (i, j) are overlapped in precedence mode m
p_j	Number of execution modes of activity j
m_{ijp}	Precedence mode of the couple (i, j) in execution mode p (of activity j)
β_{ijp}	Overlap duration between activities i and j in execution mode p (of activity j)
μ_{ijp}	Rework in activity j in execution mode p (of activity j)
ρ_{ijp}	Amount of time for coordination and communication when (i, j) are overlapped in execution mode p (of activity j)
m'_{ijp}	Precedence mode of the couple (i, j) in execution mode p (of activity i)
β'_{ijp}	Overlap duration between activities i and j in execution mode p (of activity i)
μ'_{ijp}	Rework in activity j in execution mode p (of activity i)
ρ'_{ijp}	Amount of time for coordination and communication when (i, j) are overlapped in execution mode p (of activity i)

Table 4.1: Symbols and definitions (continued)

Symbol	Definition
μ_{jp}	Rework in activity j in execution mode p
δ_{jp}	Amount of time for coordination and communication in activity j in execution mode p
CR_{jp}	Rework cost of activity j in execution mode p
CC_{jp}	Cost for coordination and communication of activity j in execution mode p
Co	Opportunity cost (cost of increasing/decreasing the makespan by one unit of time)
D	Project due date
T	Upper bound of the project makespan
C_{lim}	Upper bound of the total overlapping cost
$t = 0,\ldots, T$	Periods
EF_j, LF_j	Earliest and latest possible finish times of activity j

- Information exchange between overlapped activities require non-negligible time and cost.

When two activities are overlapped, frequent information exchange allows the downstream team to be aware of the latest upstream change. For each time information is exchanged, upstream and downstream teams set a cross-functional team meeting and discuss the latest changes for downstream incorporation (Lin et al., 2010; Loch & Terwiesch, 1998).

- For each couple of activities, overlapping is restricted to a finite number of feasible amounts of overlap, corresponding to overlapping modes.

Scheduling is performed in practice on a period-by-period basis (i.e., hour, day and week): resource availabilities and allocations are estimated per period, while activity durations are discrete multiples of one period (Hartmann, 1999). In addition, activity progress is measured in practice according to the completion of internal milestones which corresponds to important events, such as design criteria frozen, detailed design completed, drawings finalised, or any activity deliverables. This preliminary information is issued at intermediate points and used as input for a downstream activity. Therefore, the start time of an overlapped downstream activity is restricted to a finite number of instants corresponding to upstream activities' milestones which constitutes different feasible modes for overlapping. These overlapping modes can also been seen as different overlapping strategies: no overlapping, conservative overlapping and aggressive overlapping, etc...

- The problem is formulated and solved in a deterministic environment and all information required for the scheduling of the project is known in advance.

As assumed in Liberatore and Pollack-Johnson (2006) and Gerk and Qassim (2008), the problem is formulated in a deterministic environment. In particular, we assume that, for each couple of overlappable activities and for each overlapping mode, the total rework duration and cost, and the communication/coordination duration and cost, are constant and preliminary known. These parameters can be derived from models of overlapping process presented below when historical data are available.

We denote by A and P the sets of couples of overlappable and non-overlappable activities, respectively. For each activity j, Po_j and Pn_j represent the set of overlappable and non-overlappable predecessors, respectively, while So_j and Sn_j denote the set of overlappable and non-overlappable successors, respectively. The set of precedence constraints, E, the set of immediate predecessors, P_j, and successors, S_j, are defined by:

$$E = A \cup P \tag{1}$$

$$P_j = Po_j \cup Pn_j, \;\; S_j = So_j \cup Sn_j, \forall j \in S \tag{2}$$

The model of the overlapping process, the overlapping modes, the activity modes and the cost and makespan optimisation models are presented below.

4.2.1 Model of the overlapping process

Figure 4.1 shows the overlapping process of two activities (i, j) in A. The downstream activity j starts with preliminary inputs from the upstream activity i. The amount of overlap, α_{ij}, is expressed as a fraction of the downstream activity's duration. As the upstream activity proceeds, its information evolves to its final form and is released to the downstream activity j at its completion. This approach implies a more frequent exchange of evolving information during the overlapping process. The expected total duration for communication and coordination due to overlapping is denoted by σ_{ij}, with $\sigma_{ij} \geq 0$. Additional rework is often necessary to accommodate the changes in the upstream information in the downstream development. The expected duration

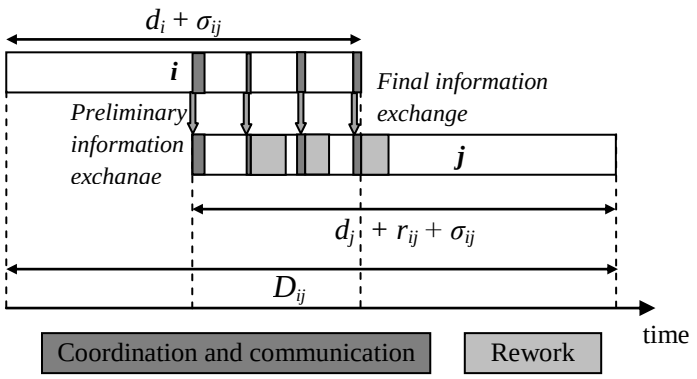

Figure 4.1: Overlapping process of two activities

of the sum of rework is denoted by r_{ij}, with $r_{ij} \geq 0$. The total amount of time required to execute both activities, D_{ij}, is expressed as follows:

$$D_{ij} = d_i + d_j \cdot (1 - \alpha_{ij}) + r_{ij} + \sigma_{ij} \tag{3}$$

If $d_i \leq d_j$, the amount of overlap is bounded by the fraction d_i / d_j in order to prevent the downstream activity to start before the upstream activity. If overlapping was not applied, the total amount of time required to execute both activities would be $D_{ij} = d_i + d_j$.

Several authors have explored the behaviour and interactions of two overlapped activities and have quantified the amount of rework and the communication/coordination duration as functions of the amount of overlap (Lin et al., 2010; Loch & Terwiesch, 1998; Roemer et al., 2000). They highlighted that the total duration of rework is a convex increasing function of the amount of overlap. In addition, when information exchange is considered, the time spent for coordination/communication is concave or convex with respect to the amount of overlap (Loch & Terwiesch, 1998). Therefore the optimal overlap amount is not necessarily the maximum feasible amount of overlap, and the time to complete both activities can be convex, concave or concave-convex.

The overlapping costs have also received much attention in order to formulate the cost minimisation and the gain maximisation problems. They are composed of the cost of rework and the communication/coordination cost. The cost of rework is usually considered as a linear function of the rework duration with or without a fixed cost (Gerk & Qassim, 2008; Lin et al., 2010; Roemer & Ahmadi, 2004; Roemer et al., 2000). For example, Roemer et al. (2000) argued

that the rework cost corresponds to hours of engineering. Similarly, Lin et al. (2010) assumed that the communication/coordination cost is proportional to the amount of time spent for communication/coordination.

However, the models proposed in the literature for a whole project with several overlappable couples of activities consider a simplistic linear relation between the rework and the amount of overlap (Cho & Eppinger, 2005; Gerk & Qassim, 2008; Liberatore & Pollack-Johnson, 2006). We propose in the next sections an optimisation model that both relaxes this assumption and encompasses any overlapping models proposed so far for two activities.

4.2.2 Precedence and overlapping modes

Overlapping is assumed to be defined for discrete values of overlap durations. Each discrete value constitutes an overlapping mode characterised by rework duration and cost, and communication/coordination duration and cost. It is important to note that these parameters are not required to be linear with the amount of overlap and that they may be null or may outweigh the time saving for some overlapping modes.

Overlapping modes can be generalised to precedence modes in order to describe all precedence constraints. For each couple of precedence constraints $i \rightarrow j$, there exists at least one precedence mode that corresponds to a basic finish-to start relation without overlapping. When $(i, j) \in A$, there exist additional precedence modes associated with the different overlapping strategies. The precedence modes can be expressed as presented in Table 4.2 and Table 4.3. When activities i and j are overlappable, they can be either overlapped ($m = 2, \ldots, m_{ij}$) or sequentially performed ($m = 1$). As depicted in Figure 4.2, the precedence constraints on the finish time of activities i and j will defer depending on the precedence mode: when not overlapped, the downstream activity start time is superior or equal to the upstream activity finish time, whereas the downstream activity start time is equal to the upstream activity finish time minus one of the feasible overlap duration in the case of overlapping.

Table 4.2: Precedence modes for a non-overlappable couple of activities (i, j) in P

Overlapping mode of couple (i, j), m	**Amount of overlap, α_{ijm}**	**Rework duration, r_{ijm}**	**Coordination/ communication duration, σ_{ijm}**	**Rework cost, Cr_{ijm}**	**Coordination/ communication cost, Cc_{ijm}**
1	0	0	0	0	0

Table 4.3: Overlapping modes for an overlappable couple of activities (i, j) in A

Overlapping mode of couple (i, j), m	**Amount of overlap, α_{ijm}**	**Rework duration, r_{ijm}**	**Coordination/ communication duration, σ_{ijm}**	**Rework cost, Cr_{ijm}**	**Coordination/ communication cost, Cc_{ijm}**
1	0	0	0	0	0
2	α_{ij2}	r_{ij2}	σ_{ij2}	$Cr_{ij}2$	Cc_{ij2}
⋮	⋮	⋮	⋮	⋮	⋮
m_{ij}	$\alpha_{i,j,mij}$	$r_{i,j,mij}$	$\sigma_{i,j,mij}$	$Cr_{i,j,mij}$	$Cc_{i,j,mij}$

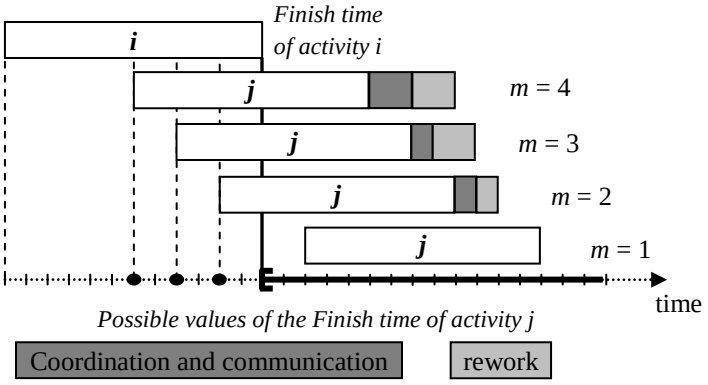

Figure 4.2: Precedence constraints depending on the overlapping modes m

4.2.3 Multiple overlapping and activity modes

We assume that there is no restriction concerning the number of overlappable predecessors. Consider the case of a activity j with two upstream activities, denoted by i_1 and i_2: if both couples (i_1, j) and (i_2, j) are overlapped, the amount of rework in downstream activity is between the maximum of single reworks and the sum of them, depending on the duplicate rework, as stated in Cho and Eppinger (2005). Without loss of generality, the latter is considered in the model.

Similarly the amount of time spent for communication/coordination in activity *j* is assumed to be the sum of the communication/coordination durations with its overlapped predecessors and successors.

As each activity can have several overlappable or non-overlappable predecessors and successors, we introduce the notion of execution modes associated to activities. Each activity mode represents a combination of possible precedence modes of an activity with its overlappable or non-overlappable predecessors and successors. The set of activity modes $\{1, \ldots, p_j\}$ for each activity is generated by a full factorial design of the precedence modes with its predecessors and successors. Table 4.4, Table 4.5 and Table 4.6 present the activity modes in different situations.

The parameters m_{ijp}, β_{ijp}, μ_{ijp} and ρ_{ijp} denote the overlapping mode, the amount of overlap, the rework duration and the communication/coordination duration of the couple (i, j) in activity mode $p = 1, \ldots, p_j$ (i.e., the activity modes of the downstream activity *j*), respectively. The same symbols with the prime symbol represent the same definitions express in activity modes of the upstream activity *i*. The parameters μ_{jp}, δ_{jp}, CR_{jp} and CC_{jp} are the total rework duration, the sum of communication/coordination duration with overlappable predecessors and successors, the total

Table 4.4: Activity modes of activity *j* in the case of non-overlappable predecessors and successors

p	$\forall i \in Pn_j$				$\forall k \in Sn_j$				μ_{jp}	δ_{jp}	CR_{jp}	CC_{jp}
	m_{ijp}	β_{ijp}	μ_{ijp}	ρ_{ijp}	m'_{jkp}	β'_{jkp}	μ'_{jkp}	ρ'_{jkp}				
1	1	0	0	0	1	0	0	0	0	0	0	0

Table 4.5: Activity modes of activity *j* in the case of one overlappable predecessor (with four overlapping modes) and no overlappable successors

p	$\forall i \in Po_j$				$\forall i' \in Pn_j$				$\forall k \in Sn_j$				μ_{jp}	δ_{jp}	CR_{jp}	CC_{jp}
	m_{ijp}	β_{ijp}	μ_{ijp}	ρ_{ijp}	$m_{i'jp}$	$\beta_{i'jp}$	$\mu_{i'jp}$	$\rho_{i'jp}$	m'_{jkp}	β'_{jkp}	μ'_{jkp}	ρ'_{jkp}				
1	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
2	2	α_{ij2}	r_{ij2}	σ_{ij2}	1	0	0	0	1	0	0	0	r_{ij2}	σ_{ij2}	Cr_{ij2}	Cc_{ij2}
3	3	α_{ij3}	r_{ij3}	σ_{ij3}	1	0	0	0	1	0	0	0	r_{ij3}	σ_{ij3}	Cr_{ij3}	Cc_{ij3}
4	4	α_{ij4}	r_{ij4}	σ_{ij4}	1	0	0	0	1	0	0	0	r_{ij4}	σ_{ij4}	Cr_{ij4}	Cc_{ij4}

Table 4.6: Activity modes of activity *j* in the case of one overlappable predecessor (with three overlapping modes) and one overlappable successor (with three overlapping modes)

p	$\forall i \in Po_j$				$\forall i' \in Pn_j$				$\forall k \in So_j$				$\forall k' \in Sn_j$				μ_{jp}	δ_{jp}	CR_{jp}	CC_{jp}
	m_{iijp}	β_{iijp}	μ_{iijp}	ρ_{iijp}	$m_{i'ijp}$	$\beta_{i'ijp}$	$\mu_{i'ijp}$	$\rho_{i'ijp}$	m'_{jkp}	β'_{jkp}	μ'_{jkp}	ρ'_{jkp}	$m'_{jk'p}$	$\beta'_{jk'p}$	$\mu'_{jk'p}$	$\rho'_{jk'p}$				
1	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
2	1	0	0	0	1	0	0	0	2	α_{jk2}	r_{jk2}	σ_{jk2}	1	0	0	0	r_{jk2}	σ_{jk2}	0	Cc_{jk2} / 2
3	1	0	0	0	1	0	0	0	3	α_{jk3}	r_{jk3}	σ_{jk3}	1	0	0	0	r_{jk3}	σ_{jk3}	0	Cc_{jk3} / 2
4	2	α_{ij2}	r_{ij2}	σ_{ij2}	1	0	0	0	1	0	0	0	1	0	0	0	r_{ij2}	σ_{ij2}	Cr_{ij2}	Cc_{ij2} / 2
5	2	α_{ij2}	r_{ij2}	σ_{ij2}	1	0	0	0	2	α_{jk2}	r_{jk2}	σ_{jk2}	1	0	0	0	$r_{ij2} + r_{jk2}$	$\sigma_{ij2} + \sigma_{jk2}$	Cr_{ij2}	$(Cc_{ij2} + Cc_{jk2})$ / 2
6	2	α_{ij2}	r_{ij2}	σ_{ij2}	1	0	0	0	3	α_{jk3}	r_{jk3}	σ_{jk3}	1	0	0	0	$r_{ij2} + r_{jk3}$	$\sigma_{ij2} + \sigma_{jk3}$	Cr_{ij2}	$(Cc_{ij2} + Cc_{jk3})$ / 2
7	3	α_{ij3}	r_{ij3}	σ_{ij3}	1	0	0	0	1	0	0	0	1	0	0	0	r_{ij3}	σ_{ij3}	Cr_{ij3}	Cc_{ij3}
8	3	α_{ij3}	r_{ij3}	σ_{ij3}	1	0	0	0	2	α_{jk2}	r_{jk2}	σ_{jk2}	1	0	0	0	$r_{ij3} + r_{jk2}$	$\sigma_{ij3} + \sigma_{jk2}$	Cr_{ij3}	$(Cc_{ij3} + Cc_{jk2})$ / 2
9	3	α_{ij3}	r_{ij3}	σ_{ij3}	1	0	0	0	3	α_{jk3}	r_{jk3}	σ_{jk3}	1	0	0	0	$r_{ij3} + r_{jk3}$	$\sigma_{ij3} + \sigma_{jk3}$	Cr_{ij3}	$(Cc_{ij3} + Cc_{jk3})$ / 2

rework cost and the total communication/coordination cost of activity j in mode $p = 1, \ldots, p_j$. Note that the communication/coordination cost associated with a couple (i, j) in A are equally split between activity i and j.

4.3 Performance optimisation models

Each activity j in S must finish within the time window $\{EF_j, \ldots, LF_j\}$ with respect to the precedence relations, the overlapping opportunities and the activity durations. It can be derived from the traditional forward recursion and backward recursion algorithms considering that the project must start at time 0 and that T constitutes an upper bound of the project's makespan. We define the decision variables (i.e., the finish times and the overlapping modes) as follows:

$$X_{jtp} = \begin{cases} 1 & \text{if activity } j \text{ is executed in mode } p \text{ and finishes at time } t \\ 0 & \text{otherwise} \end{cases}$$

$$\forall j \in S, \forall t \in [0, T] \text{ and } \forall p \in [1, p_j] \tag{4}$$

$$Y_{ij} = \begin{cases} 1 & \text{if activity } i \text{ and } j \text{ are overlapped} \\ 0 & \text{otherwise} \end{cases}, \ \forall j \in S, \ \forall i \in Po_j$$

The decision on the activity modes can be classed into three cases. On the one hand, if activity j is not overlappable with any immediate predecessor or successor, the decision is simply not to overlap. On the other hand, if activity j is overlappable, this activity can be either overlapped with at least one of its overlappable predecessors or successors ($p > 1$) or not overlapped ($p = 1$). Note that Y_{ij} is an additional binary variable introduced to linearise the optimisation problems. The resource-constrained scheduling problem with overlapping is formulated in the next section with the objective of maximising the project gain. Then, we present a variation of this problem with the objective of minimising the project makespan.

4.3.1 Project gain maximisation problem

The resource-constrained scheduling problem of maximising the project gain with overlapping can be formulated with a linear 0-1 integer programming model as follows:

$$\text{Maximize} \quad Co\cdot\left(D-\sum_{p=1}^{p_{n+1}}\sum_{t=EF_{n+1}}^{LF_{n+1}} t\cdot X_{n+1,t,p}\right)-\sum_{j=2}^{n}\sum_{p=1}^{p_j}\sum_{t=EF_j}^{LF_j}\left(CR_{jp}+CC_{jp}\right)\cdot X_{jtp} \tag{5}$$

Subject to

$$\sum_{p=1}^{p_i}\sum_{t=EF_i}^{LF_i}\left(t-\rho_{ijp}\right)\cdot X_{itp}\le\sum_{p=1}^{p_j}\sum_{t=EF_j}^{LF_j}\left[t-d_j\cdot\left(1-\beta_{ijp}\right)-\left(\mu_{jp}+\delta_{jp}\right)\right]\cdot X_{jtp}\ ,\ \forall j\in S\ ,\ \forall i\in P_j \tag{6}$$

$$\sum_{p=1}^{p_j}\sum_{t=EF_j}^{LF_j}\beta_{ijp}\cdot X_{itp}\le Y_{ij}\ ,\ \forall j\in S\ ,\ \forall i\in Po_j \tag{7}$$

$$\sum_{p=1}^{p_i}\sum_{t=EF_i}^{LF_i}\left(t-\rho_{ijp}\right)\cdot X_{itp}\ge\sum_{p=1}^{p_j}\sum_{t=EF_j}^{LF_j}\left[t-d_j\cdot\left(1-\beta_{ijp}\right)-\left(\mu_{jp}+\delta_{jp}\right)\right]\cdot X_{jtp}-T\cdot\left(1-Y_{ij}\right),$$
$$\forall j\in S\ ,\ \forall i\in Po_j \tag{8}$$

$$\sum_{j=2}^{n}\left[R_{jk}\cdot\left(\sum_{p=1}^{p_j}\sum_{b=t}^{t+d_j-1+\mu_{jp}+\delta_{jp}}X_{jbp}\right)\right]\le R_k\ ,\ \forall k\in R\ ,\ \forall t\in\left[0,T\right] \tag{9}$$

$$\sum_{p=1}^{p_i}\sum_{t=EF_i}^{LF_i}t\cdot X_{itp}\le\sum_{p=1}^{p_j}\sum_{t=EF_j}^{LF_j}t\cdot X_{jtp}\ ,\ \forall j\in S\ ,\ \forall i\in Po_j \tag{10}$$

$$\sum_{p=1}^{p_i}\sum_{t=EF_i}^{LF_i}\beta'_{ijp}\cdot X_{itp}=\sum_{p=1}^{p_j}\sum_{t=EF_j}^{LF_j}\beta_{ijp}\cdot X_{jtp}\ ,\ \forall j\in S\ ,\ \forall i\in Po_j \tag{11}$$

$$\sum_{p=1}^{p_j}\sum_{t=EF_j}^{LF_j}X_{jtp}=1,\ \forall j\in S \tag{12}$$

$$X_{jtp}=\{0,1\},\forall j\in S\ ,\forall t\in\left[0,T\right]\text{and}\ \forall p\in\left[1,p_j\right] \tag{13}$$

$$Y_{ij}=\{0,1\},\ \forall j\in S\ ,\ \forall i\in Po_j \tag{14}$$

The objective function (5) maximises the project gain, which is composed of the opportunity cost for finishing earlier or later than a target time, D (i.e., the project makespan without overlapping or any project due date), and the overlapping costs (rework and communication/coordination costs). Constraints (6), (7) and (8) reflect the precedence and overlapping constraints presented in Figure 4.2. Constraints (6) represent the finish-to-start precedence constraints, with a negative lead time in the case of overlapping. According to constraints (7), if two overlappable activities (i, j) are overlapped, then Y_{ij} = 1 and thus the union of constraints (6) and (8) ensure that the downstream activity must start at the upstream activity finish time minus one of the feasible overlap duration. If activities (i, j) are not overlapped, then Y_{ij} is unrestricted and constraints (8) are not restrictive. Constraints (9) define the resource constraints. Constraints (10) state that the downstream activity of a couple of overlappable activities cannot finish before the upstream activity's finish time. Constraints (11) state that the activity modes of two overlappable activities are such that the overlapping modes are the same for both activities. Constraints (12) ensure that each activity is associated with one activity mode and one finish time. Finally, constraints (13) and (14) define the aforementioned binary decision variables.

4.3.2 Project makespan minimisation problem

The project makespan minimisation problem can be formulated by replacing the objective function (5) by the following objective function:

$$\text{Minimize} \quad \sum_{p=1}^{p_{n+1}} \sum_{t=EF_{n+1}}^{LF_{n+1}} t \cdot X_{n+1,t,p} \tag{15}$$

The objective function (15) minimises the finish time of the dummy sink activity. In addition, we introduce the following constraint to ensure that the total overlapping cost will not exceed a given value:

$$\sum_{j=2}^{n} \sum_{p=1}^{p_j} \sum_{t=EF_j}^{LF_j} \left(CR_{jp} + CC_{jp} \right) \cdot X_{jtp} \leq C_{li} m \tag{16}$$

The project gain maximisation and makespan minimisation problems allow to study the time-cost trade-offs between project duration and overlapping costs with resource constraints. To our

knowledge, such scheduling problems with and overlapping has only been treated by Liberatore and Pollack-Johnson (2006) and Gerk and Qassim (2008). However, these authors used a simplified overlapping model with a linear relation between rework and overlap duration and they assumed that information exchange is instantaneous and costless. Our formulation allows any overlapping process to be applied, with the condition that overlapping can only be executed at feasible modes and information exchange between overlapped activities require non-negligible time and cost.

4.4 Illustrative example

We consider a project instance generated by Kolisch and Sprecher (1997) composed of 30 non-dummy activities and four renewable resources (see Table 4.7). The availability of the resources are set to $R_k = 20$, $k = 1, ..., 4$. As no overlapping was defined in the original instance, the additional overlapping data have been generated. Fifteen couples of overlappable activities with three overlapping modes have been considered, as depicted in Table 4.8. Note that overlapping does not necessarily entail reworks and additional coordination and communication. For example, $\sigma_{ijm} = r_{ijm} = 0$ when $(i, j) = (2, 3)$ and $m = 2$. As assumed in Roemer et al. (2000), Roemer and Ahmadi (2004) and Lin et al. (2010) , the rework and communication/coordination costs are considered as linear functions of the time spent on rework and communication (i.e., \$200 per unit time for each resource). Note that any other function could be used in our model. The activity modes can easily be derived from Table 4.8. For the sake of conciseness, we only present in Table 4.9 the activity modes of activity 8 as an example. Finally, the opportunity cost is set to \$5,000 per unit time, while the project due date is set to the makespan found without overlapping. The illustrative case was implemented in AMPL Studio v1.6.j and solved with Cplex 12.2.

Table 4.7: Project data composed of 30 non-dummy activities

Activity j	**Duration** d_j	**Resource consumption**				**Set of non-overlappable predecessors,** Pn_j	**Set of overlappable predecessors,** Po_j
		R_{j1}	R_{j2}	R_{j3}	R_{j4}		
1	0	0	0	0	0	{}	{}
2	14	1	6	0	0	{1}	{}
3	5	5	2	0	4	{}	{2}
4	5	0	0	0	3	{3}	{}
5	12	4	8	6	8	{2}	{}
6	11	0	7	0	0	{}	{3}
7	5	7	0	0	8	{3}	{}
8	6	0	0	6	10	{}	{7}
9	8	6	3	8	0	{2}	{}
10	12	0	1	3	6	{}	{8}
11	14	7	0	7	0	{4,8}	{}
12	15	0	0	0	8	{5}	{10}
13	13	0	0	2	5	{}	{10}
14	15	5	4	0	7	{}	{11}
15	5	4	0	0	0	{4, 7, 9}	{}
16	5	2	0	0	0	{13}	{}
17	10	1	0	0	3	{10}	{11}
18	12	4	1	0	4	{4}	{6}
19	12	0	0	10	3	{13, 15, 18}	{}
20	9	4	6	0	0	{7, 18}	{}
21	7	8	2	9	0	{}	{13, 20}
22	6	3	0	7	0	{8, 15}	{}
23	14	0	0	0	6	{5, 9, 21}	{}
24	7	8	0	0	0	{12, 19, 23}	{}
25	12	8	0	0	4	{16, 17}	{24}
26	8	0	8	8	0	{9}	{}
27	7	0	1	3	7	{14, 17, 21}	{}
28	15	4	0	0	2	{15, 26, 27}	{}
29	15	0	8	0	6	{11, 21}	{19}
30	8	3	0	7	8	{5, 29}	{28}
31	8	0	0	0	5	{22, 30}	{25}
32	0	0	0	0	0	{31}	{}

Table 4.8: Overlapping data for the couples of overlappable activities

activities (i, j)	mode m	α_{ijm}	r_{ijm}	σ_{ijm}	Cr_{ijm} ($)	Cc_{ijm} ($)	activities (i, j)	mode m	α_{ijm}	r_{ijm}	σ_{ijm}	Cr_{ijm} ($)	Cc_{ijm} ($)
(2, 3)	1	0	0	0	0	0	(11, 17)	1	0	0	0	0	0
	2	0.4	0	0	0	0		2	0.3	1	0	800	0
	3	0.8	1	0	2200	0		3	0.6	2	1	1600	3600
(3, 6)	1	0	0	0	0	0	(13, 21)	1	0	0	0	0	0
	2	2/11	1	0	1400	0		2	2/7	0	0	0	0
	3	4/11	1	1	1400	3600		3	4/7	2	1	7600	5200
(6, 18)	1	0	0	0	0	0	(19, 29)	1	0	0	0	0	0
	2	5/12	1	1	1800	3200		2	4/15	1	0	2800	0
	3	8/12	3	2	5400	6400		3	8/15	2	1	5600	5400
(7, 8)	1	0	0	0	0	0	(20, 21)	1	0	0	0	0	0
	2	1/3	1	0	3200	0		2	3/7	1	0	3800	0
	3	2/3	2	1	6400	6200		3	5/7	2	1	7600	5800
(8, 10)	1	0	0	0	0	0	(24, 25)	1	0	0	0	0	0
	2	0.25	1	0	2000	0		2	0.25	1	1	2400	4000
	3	5/12	2	0	4000	0		3	5/12	2	1	4800	4000
(10, 12)	1	0	0	0	0	0	(25, 31)	1	0	0	0	0	0
	2	4/15	1	0	1600	0		2	0.25	1	0	1000	0
	3	8/15	2	2	3200	7200		3	0.625	2	1	2000	3400
(10, 13)	1	0	0	0	0	0	(28, 30)	1	0	0	0	0	0
	2	4/13	1	0	1400	0		2	0.375	1	0	3600	0
	3	9/13	2	2	2800	6800		3	0.75	1	1	3600	4800
(11, 14)	1	0	0	0	0	0							
	2	1/3	2	0	6400	0							
	3	0.6	3	2	9600	12000							

Table 4.9: Activity modes of activity 8

p	Overlappable predecessor, $i = 7$				Overlappable successor, $k = 10$				Non-overlappable successors, $k' = 11, 22$				μ_{8p}	δ_{8p}	CR_{8p} ($)	CC_{8p} ($)
	m_{78p}	β_{78p}	μ_{78p}	ρ_{78p}	$m'_{8,10,p}$	$\beta'_{8,10,p}$	$\mu'_{8,10,p}$	$\rho'_{8,10,p}$	$m'_{jk'p}$	$\beta'_{jk'p}$	$\mu'_{jk'p}$	$\rho'_{jk'p}$				
1	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
2	1	0	0	0	2	0.25	1	0	1	0	0	0	0	0	0	0
3	1	0	0	0	3	5/12	2	0	1	0	0	0	0	0	0	0
4	2	1/3	1	0	1	0	0	0	1	0	0	0	1	0	3200	0
5	2	1/3	1	0	2	0.25	1	0	1	0	0	0	1	0	3200	0
6	2	1/3	1	0	3	5/12	2	0	1	0	0	0	1	0	3200	0
7	3	2/3	2	1	1	0	0	0	1	0	0	0	2	1	6400	3100
8	3	2/3	2	1	2	0.25	1	0	1	0	0	0	2	1	6400	3100
9	3	2/3	2	1	3	5/12	2	0	1	0	0	0	2	1	6400	3100

4.4.1 Project gain maximisation and makespan minimisation problems

The gain maximisation and the makespan minimisation problems are investigated in this section. The latter is formulated without any upper bound for the overlapping cost. For each optimisation criterion, the scheduling problems with or without resource constraints and with or without overlapping modes are presented.

Table 4.10 highlights that a significant reduction of the optimal makespan is obtained with overlapping in our illustrative example: 11.11% and 15.53% with resource and without resource constraints, respectively. For the problem without resource constraints and with overlapping modes (case 3), almost all overlappable activities are overlapped (13 out of 15) and most of them are overlapped at their local minimum (10 out of 13). The project gain obtained with the makespan minimisation problem with resource constraints is slightly positive. By contrast, the optimal makespan and the corresponding gain found in cases 1 and 3 shows that minimising the makespan with overlapping can result in a lower gain, as the overlapping costs may exceed the opportunity cost.

The overlapping modes of the optimal makespan obtained for both the makespan minimisation and the gain maximisation problems are detailed in Table 4.11. When resource constraints are

considered, overlapping is less performed than without resource constraints in our illustrative

Table 4.10: Effects of resource constraints and overlapping on the optimal project makespan

Case	Resource constraints	Overlapping modes	Number of overlappable activities	Number of overlapped activities	Optimal makespan	Corresponding gain	CPU's time (s)
1	No	No	0	0	103	25000	0.09
2	Yes	No	0	0	108	0	0.36
3	No	Yes	15	13	87	23000	2.09
4	Yes	Yes	15	7	96	3000	1708.05

Table 4.11: Effects of resource constraints and overlapping on the optimal project gain

couple of overlappable activities	Overlapping mode			
	Makespan minimization problem		Gain maximization problem	
	Case 3	Case 4	Case 3	Case 4
(2, 3)	3	3	3	2
(3, 6)	2	1	2	1
(6, 18)	2	2	2	1
(7, 8)	3	1	2	1
(8, 10)	3	1	2	2
(10, 12)	1	1	1	1
(10, 13)	3	3	2	2
(11, 14)	2	1	2	1
(11, 17)	1	1	1	1
(13, 21)	2	2	2	1
(19, 29)	3	1	1	1
(20, 21)	2	3	1	1
(24, 25)	3	2	1	1
(25, 31)	2	1	2	1
(28,30)	3	3	1	3

example. As expected, overlapping lead to additional workload and to more resource consumptions. Overlapping is thus less attractive and less than half of the set of overlappable activities are overlapped with resource constraints and overlapping modes (case 4). This confirms that overlapping and resource constraints are closely interrelated.

The results obtained for the project gain maximisation problem in Table 4.12 show that the optimal gains with overlapping modes are significantly better than those without overlapping, and better than the gain obtained for the project makespan minimisation. However, the

Table 4.12: Effects of resource constraints and overlapping on the optimal project gain

Case	Resource constraints	Overlapping modes	Number of overlappable activities	Number of overlapped activities	Optimal makespan	Corresponding gain	CPU's time (s)
1	No	No	0	0	25000	103	0.09
2	Yes	No	0	0	0	108	0.36
3	No	Yes	15	9	62400	91	2.09
4	Yes	Yes	15	4	38200	99	1251.41

corresponding makespan is not as good as for the project makespan minimisation. Indeed, the optimal schedules in terms of project gain lead to a 8.33% and a 12.50% reduction of the makespan with and without resource constraints, respectively.

Tables 4.10 and 4.12 also reveal that overlapping modes significantly increase the computational time required to solve the optimisation problems for this illustrative example, as it adds further complexity to the already complex case of resource-constrained scheduling problem, which is known to be a NP-hard optimization problem (Herroelen, 2005).

The optimal schedules obtained for the gain maximisation and the makespan minimisation problems differ in terms of resulting gain and makespan for this illustrative example. Minimising the project makespan requires overlapping several couples of activities, which entails additional rework and communication/coordination costs. In order to conciliate these two contradictory targets, we propose to investigate the time-cost trade-off in the next section.

4.4.2 Time-cost trade-off analysis

This section aims to provide a deeper insight into the time-cost trade-off involving the opportunity cost for finishing the project earlier or later and the overlapping costs. We also investigate the effects of the resource constraints on the time-cost trade-off by varying the resource availabilities to upper and lower values. Figure 4.3 presents the effects of overlapping on the project performances. These results were derived from the project gain maximisation

programming model described in Section 4.3.1. An additional constraint on the project makespan was introduced in order to compute the project gain maximisation problem for different values of the project makespan. The corresponding overlapping costs are depicted in Figure 4.4.

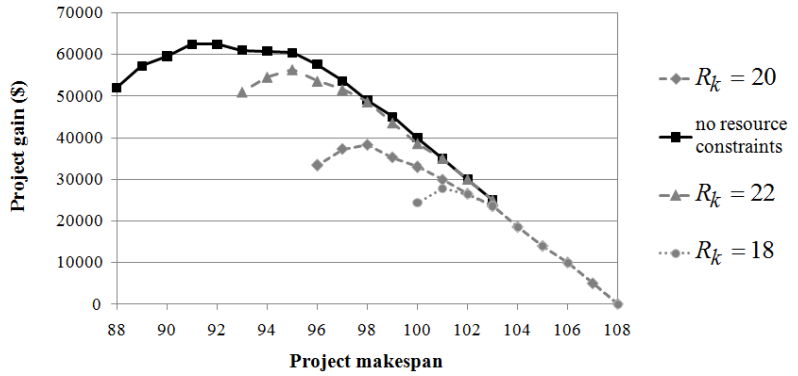

Figure 4.3: The effects of overlapping on the project performances

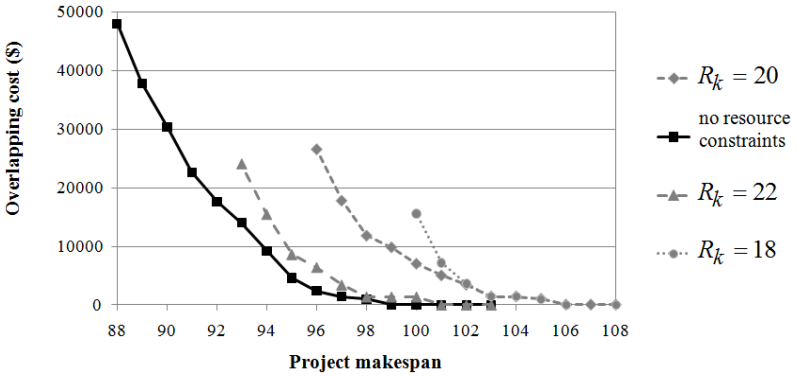

Figure 4.4: Minimum overlapping cost as a function of the project makespan

Figure 4.4 highlights that, for this illustrative example, the cost of decreasing the project makespan by one unit time increases as the project makespan decreases. Indeed, the overlapping data generated in the numerical example are such that the rework cost and duration, as well as the communication/coordination cost and duration are non-decreasing with respect to the amount of overlap for each couple of overlappable activities (see Table 4.8). In addition, more activities are executed in parallel and more and more couples of overlappable activities must be overlapped to gain one unit time when the project schedule is compressed. As shown in the previous section,

the scarcer the resources, the more restricted the set of potential overlapped activities required to reduce the project makespan by one unit. Consequently, it becomes more expensive to overlap and reduce the project makespan when the resource availability is lower.

As the overlapping costs required to reduce the project makespan by one unit time increase, it may outweigh the opportunity cost for this illustrative example. Figure 4.3 shows that project gain decreases when activities are overlapped beyond a certain point.This explains why the optimal makespan obtained with the project gain maximization problem in the previous section differs from the minimal makespan. Depending on the resource availability and how much the project managers and planners are willing to pay to complete the project earlier, the time-cost trade-off analysis presented in this section should help them to choose appropriate overlapping strategies.

4.5 Conclusions and discussion

Overlapping activities is one of the most applied strategies used to accelerate a project. Overlapping entails that downstream activities start before the information they require is available in a finalised form. However, additional communication and coordination and additional workload that may be required to accommodate the information changes transmitted by upstream activities are often ignored in practice. In spite of all research efforts accomplished in evaluating the relation between the amount of overlap and the reworks for two activities without resource constraints (Krishnan et al., 1997; Lin et al., 2010; Loch & Terwiesch, 1998; Roemer et al., 2000), only few papers have incorporated overlapping in the RCPSP of whole projects (Cho & Eppinger, 2005; Gerk & Qassim, 2008). In addition, these papers studied simplified linear rework model that are not realistic and considered instantaneous and costless information exchange.

The main contribution of this paper is to present a linear integer programming model for the project gain maximisation and the project makespan minimisation problems with overlapping modes and resource constraints. The development of mathematical formulations is important not only to take into account the characteristics of applications arising from practice, such as overlapping and rework and communication/coordination costs, but also to raise problem structures that may be used in designing new heuristics. Furthermore, our model makes it

possible to quantify the effect of overlapping activities on the project performances in terms of cost and time and to assist project managers and planners to choose the most suitable overlapping strategy.

Nonetheless, we would like to point out several limitations of our approach, and suggest some possible directions for future research. First, the proposed problem formulation shares similarities with the traditional multi-mode resource constraint scheduling problem (MRCPSP), as activities can be executed in several modes with different durations. Exact dedicated procedures could be developed and, considering the limit of exact solution procedure encountered with MRCSPSP (Hartmann & Briskorn, 2010; Herroelen et al., 1998; Kolisch & Padman, 2001), we can anticipate that solving the RCPSP with overlapping modes for larger projects will require the use of metaheuristics or heuristics. These heuristics could use the structure revealed by the basic model proposed in this paper.

Second, our model requires the estimation of the reworks and the time spent for communication and coordination. If some methods have been proposed in the literature for that purpose when organisations have experience with similar projects (Krishnan et al., 1997; Lin et al., 2010; Loch & Terwiesch, 1998; Roemer et al., 2000), the problem of how to reliably estimate these data for new projects should also be investigated. In addition, we suggest to test the model with many instances in order to derive general insights and to capture the effects of different parameters, such that the number of overlappable activities, the number of precedence relations, the resource consumption and the resource availability.

Finally, the problem is formulated in a deterministic environment and does not directly address schedule risks. However, overlapping is inherently risky as it entails that downstream activities start before the information they require is available in a finalized form. We may extend the model to introduce randomness along several parameters and to consider feedbacks and iterations. As some of the most advanced approaches developed in the literature for project scheduling under uncertainty, such as proactive and reactive techniques, involve determining a baseline schedule without anticipation of uncertainty (Demeulemeester et al., 2008; Guéret & Jussien, 2008; Herroelen & Leus, 2004, 2005), the formulation in a deterministic environment proposed in this paper constitutes a first step towards the development of approaches for scheduling projects with overlapping under uncertainty.

CHAPITRE 5 ARTICLE 2: A PATH RELINKING-BASED SCATTER SEARCH FOR THE RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEM

rédigé par:

François Berthaut[3], Robert Pellerin[3], Adnène Hajji[4] et Nathalie Perrier[3]

soumis pour publication dans :

European Journal of Operational Research

date de soumission initiale: 18 septembre 2014

date de réception des commentaires du comité de lecture : 6 janvier 2015

date de soumission de la version révisée : 24 avril 2015

Abstract

Project scheduling has received a growing attention from researchers for the last decades in order to propose models and methods to tackle scheduling problems for real-size projects. In this paper, we consider the resource-constrained project scheduling problem (RCPSP), which consists in scheduling the activities in order to minimize the project duration in presence of precedence and resource constraints. As this problem is NP-hard, we propose a hybrid metaheuristic based on scatter search that involves forward-backward improvement and reversing the project network at each iteration of the scatter search. Two new mechanisms are introduced. First, a bidirectional path relinking method with a new move is used as combination method. Second, a new improvement procedure is proposed in the reference set update method. An advanced parameter tuning method based on local search is employed. The proposed method is applied to the standard

[3] Jarislowsky/SNC-Lavalin Research Chair in the Management of International Projects and CIRRELT, Department of Mathematics and Industrial Engineering, École Polytechnique de Montreal, Montreal, Canada (francois.berthaut@polymtl.ca; robert.pellerin@polymtl.ca;nathalie.perrier@polymtl.ca)

[4] Department of Operations and Decision Systems, Faculty of Business Administration, Laval University, Québec, Canada (adnene.hajji@fsa.ulaval.ca)

benchmark projects of size 30, 60 and 120 activities from the PSPLIB library and compared with the state-of-the-art heuristics of the literature. The computational results show that the proposed hybrid scatter search produces high-quality solutions in reasonable computational time and is among the best performing metaheuristics.

Keywords: Resource-constrained Project scheduling, Metaheuristics, Makespan minimization, Scatter Search, Path Relinking.

5.1 Introduction

Project scheduling consists in determining the start and end times of the project activities in presence of scarce resources and precedence relations. Since the early 1960's, this problem has attracted both researchers and practitioners. For the latter, scheduling serves major functions and is often supported in practice by a project planning software which models the project, solves the scheduling problem and provides representations for communication. For researchers, project scheduling is very attractive because the models are rich, have a variety of applications and are difficult to solve (Brucker et al., 1999). The standard project scheduling problem is the resource-constrained project scheduling problem (RCPSP), which involves the determination of a precedence- and resource-feasible schedule that minimizes the project duration. Even though the assumptions of the RCPSP do not cover all the situations that occur in practice, many more general models, variants and extensions have been developed that often use the RCPSP as starting point. For more details on the research on RCPSP, the reader is referred to the following surveys: Özdamar and Ulusoy (1995), Herroelen et al. (1998), Brucker et al. (1999), Kolisch and Hartmann (1999), Hartmann and Kolisch (2000), Kolisch and Padman (2001), Tavares (2002), Herroelen (2005), Kolisch and Hartmann (2006), Hartmann and Briskorn (2010).

The RCPSP belongs to the class of NP-hard optimization problems, and hence, is one of the most intractable problems in operations research (Blazewicz et al., 1983). Whereas surveys presented in Icmeli-Tukel and Rom (1997), Liberatore, Pollack-Johnson and Smith (2001), Liberatore and Pollack-Johnson (2003) show that more than half of projects in practice have a size larger than one hundred activities, Herroelen (2005) pointed that exact procedures have only the capability to solve small scale projects with at most sixty activities and slight resource constraints in acceptable computation effort. This limitation has motivated a growing number of academics to

develop heuristic procedures able to find good-quality schedules in reasonable computation effort for larger projects usually encountered in practical cases. In their experimental investigation of the best heuristics for the RCPSP, Hartmann and Kolisch (2000) presented the computational results of thirteen heuristics. In the updated version of Kolisch and Hartmann (2006), thirty-seven heuristics were compared. From 2006 to 2013, we identified more than forty additional heuristics for the RCPSP in the literature. The current best performing heuristics are metaheuristics, which are capable of learning. The most efficient of them do not follow classical paradigms but combine concepts of different classical metaheuristics into hybrid approaches and use advanced mechanisms such as the forward-backward improvement (FBI), the path relinking (PR) and specific representations (Herroelen, 2005; Kolisch & Hartmann, 2006).

In the last years, several researchers have proposed hybrid metaheuristics based on the scatter search framework. Conceptualized by Glover (1977), scatter search (SS) is an evolutionary method inspired from the principle that systematic designs and methods for creating new solutions afford significant benefits beyond those derived from recourse to randomization. The first application of SS for the RCPSP can be found in Valls et al. (2004), who presented a population-based approach that alternates between a forward and backward scheduling procedure to improve the resource utilization and a blend of SS and PR strategies to combine and evolve the population of solutions. Debels et al. (2006) proposed a hybrid SS that incorporates a combination method based on the principles of electromagnetism and uses FBI as improvement method. Ranjbar et al. (2009) used the same SS backbone as in Debels et al. (2006), but reversed the project network at each population generation and combined solution using PR. Mobini et al. (2009) derived an enhanced SS that employs FBI as improvement method and combines solutions with a two point cross-over operator, a PR strategy and a permutation-based operator. Chen et al. (2010) proposed an ant colony optimization that uses SS as improvement method and the resource utilization improvement procedure developed by Valls et al. (2004). Finally, Paraskevopoulos et al. (2012) developed a hybrid SS that combines solutions using a linear combination method based on events' starting times, and improves them with an adaptive iterative local search. Based on the experimental results on the PSPLIB benchmark (Kolisch & Sprecher, 1997), these SS based algorithms have been identified as being among the current best heuristics.

This paper is motivated by the attractive results obtained with SS for the RCPSP and contributes to the long-term objective of developing efficient metaheuristics that can handle large-sized projects and that can be adapted for more complex scheduling problems. We propose a new hybrid metaheuristic that relies on a SS skeleton inspired from Debels et al. (2006), Ranjbar et al. (2009) and Mobini et al. (2009). In particular, this metaheuristic involves FBI and reversing the project network at each iteration. A new improvement procedure is added into the reference set update method in order to enhance the quality or the diversity of the reference set of solutions. Moreover, the proposed combination method presents two distinctive features. It is based on a bidirectional PR, which has been identified as a promising approach for combination in SS by Glover, Kelly and Laguna (1995), Marti, Laguna and Glover (2006) and Resende, Ribeiro, Glover and Marti (2010). This PR generates solutions by exploring a path build by gradual moves between two solutions to be combined. Also, we introduce a new move that consists in moving the most distant activity from the guiding solution in the current solution of the path to its place in the guiding solution. The computational results show that the proposed hybrid scatter search produces high-quality solutions in reasonable computational time and is among the best performing metaheuristics.

The remainder of this paper is organized as follows. Section 5.2 introduces the definition of the RCPSP and the notations. Section 5.3 provides a description of the proposed hybrid SS. In Section 5.4, an advanced parameter tuning method based on local search is proposed and computational results are presented, including a comparison with the state-of-the-art heuristics from the literature. Finally, we conclude by summarizing the main results and by highlighting possible directions for further research in Section 5.5.

5.2 Resource-Constraint Project Scheduling Problem definition

The RCPSP problem can be defined as follows. A single project is composed of a set N of n activities labeled $j = 1,...,n$, and two dummy activities 0 and n+1, which are the project start and project end, respectively. We denote by d_j the duration of activity j. The activities have to be performed without preemption. Technological reasons imply that some activities must finish before others can start. These precedence relations are denoted by $i \rightarrow j$. P_j and S_j represent the set of immediate predecessors and successors of activity j, respectively. Each activity requires constant amounts of renewable resources to be performed. These resources are called renewable

because they are fully available at each period. There are K resource types and we denote by r_{ik} the usage of resource $k \in K$ per period by activity $j \in N$. The constant capacity of each resource type is represented by R_k. The two dummy activities have zero duration and no resource usage. All information is assumed to be deterministic and known in advance. The parameters are assumed to be nonnegative and integer-valued.

A schedule is defined as an assignment of start times s_j and finish times f_j to activities $j \in N$. A schedule is called precedence-feasible if $s_j \geq f_i$ for each precedence constraint $i \rightarrow j$, and resource-feasible if the consumption of resources does not exceed the resource capacity for each time period and each resource type. The RCPSP consists in determining a precedence and resource-feasible schedule so as to minimize the project duration (project makespan). The reader is referred to Herroelen et al. (1998) for a mathematical model of the RCPSP.

5.3 Hybrid Scatter Search Algorithm

Introduced by Glover (1977), SS is an evolutionary algorithm that generates new solutions by combining preserved ones. SS is composed of the following basic methods (Marti et al., 2006):

- diversification generation method: to generate an initial population of trial solutions;
- improvement: to transform a trial solution into enhanced solutions;
- reference set update method: to build and maintain a reference set of the best solutions, that is often partitioned into a set of high-quality solutions and a set of diversified solutions;
- subset generation method: to produce subsets of solutions of the reference set. The common method is to generate pairs of reference solutions;
- solution combination method: to transform a given subset of solutions produced by the subset generation method into one or more combined solutions.

SS is a very flexible methodology since each element can be implemented with different degree of sophistication. It uses strategies for intensification and diversification that have been proved effective in a variety of optimization problems (Marti et al., 2006).

In this paper, the algorithm proposed for solving the RCPSP is based on these methods and is presented in Figure 5.1. The methods involving the new mechanisms introduced in the paper, namely the reference set update method and the PR-based combination method, are further detailed in Figure 5.2 and Figure 5.3, respectively. An initial population of size *InitPop* is first generated. A reference set *RefSet* composed of two distinct sets of high-quality and diversified solutions is build from the population of solutions. These reference solutions will be evolved to form a new population. This new population is first initialized with the best current solution. The reference solutions are afterwards paired to form subsets that are combined with PR to generate

---

1. Initial population generation
   a. Generate *InitPop* activity lists with the regret-biased random sampling method by using the minimum LFT priority rule.
   b. Evaluate each *AL* with the serial SGS (forward scheduling, *rev* = 0).
   c. Apply the improvement strategy based on FBI to the best solutions.
   d. Set *init* = 1 and *New_Pop* = *InitPop*.

While (*NSched* < *NSched_limit*) do

2. Reverse the type of schedule
   a. *rev* = 1 - *rev*.
   b. Apply TO condition to each schedule of the population.
3. Reference set update
   Contruct the set of high-quality solutions $RefSet_1$ and the set of diversified solutions $RefSet_2$ from *New Pop*. $RefSet = RefSet_1 \cup RefSet_2$.
4. Generate the *Subsets* from *RefSet*
   Generate all pairs of solutions in $RefSet_1$, and all combination of one solution from $RefSet_1$ and another one from $RefSet_2$.
5. Initialize the *New Population* with the Best Solution
6. For all Subsets apply the PR-based combination method and the improvement strategy and add the new solutions to *New Pop*

End While

7. Return the best solution

Figure 5.1: The path relinking-based scatter search algorithm for the RCPSP

new solutions. These solutions are evaluated and either directly added to the new population *New_Pop* or first improved by FBI depending on their quality. The algorithm stops when the number of generated schedules reaches *NSched_limit*. At each iteration, the scheduling direction and the project network are reversed. The solution representation and decoding, and the details of each step are described below.

5.3.1 Solution representation and decoding

Usually, metaheuristic approaches for the RCPSP operate on representations of schedules rather than directly on schedules themselves (Kolisch & Hartmann, 1999). A schedule generation scheme (SGS) is then required to transform a representation into a feasible schedule. Two different SGSs are usually considered in the literature: the serial and the parallel SGS. The dominant representations are the activity list and the random key. Experimental investigations have shown that the best metaheuristics employ the serial SGS combined with the activity list representation (Hartmann & Kolisch, 2000; Kolisch & Hartmann, 2006). We consequently choose this combination of methods for the proposed SS.

The serial SGS starts from scratch and builds a feasible schedule by stepwise extension of a partial schedule. At each stage, an activity is selected from the eligible set and scheduled at the earliest precedence- and resource-feasible start time. The eligible set comprises all the activities whose predecessors are already scheduled. A representation of schedules is required to drive the selection of activities. The activity list representation makes use of a sequence of all activities $AL = (a_0,\ldots, a_{n+1})$ in which the position of an activity determines its relative priority with the other activities for being selected. We also define the vector $P = (p_0,..., p_{n+1})$ to represent the position of each activity in AL. If $i = a_h$, the activity i is in position $p_i = h$. As defined, an AL may not be precedence-feasible. In the literature, most authors use precedence-feasible activity lists in which each activity is placed in the list after all of its predecessors. Hartmann and Kolisch (2000) noted that such lists are faster to decode, since the construction of the eligible set and the selection of an activity do not need to be computed during the serial SGS procedure. Indeed, the serial SGS simply selects at each stage j the activity a_j of the precedence-feasible activity list. Consequently, any AL will be considered as precedence-feasible for the rest of the paper.

A major problem with AL representation is that a single schedule can be represented by more than one activity list. This is caused by timing anomalies and activities with identical starting times, as explained in Debels et al. (2006). Consequently, the solution space is filled with useless replications of representations, and thus, the search efficiency is typically reduced (Paraskevopoulos et al., 2012). To tackle these issues, Valls, Quintanilla and Ballestin (2003) introduced the topological order (TO) representation, which is an activity list that satisfies: $s_i < s_j$ implies $p_i < p_j$. Ties are broken by the activity index: $s_i = s_j$ and $i < j$ implies $p_i < p_j$. The TO

representation can be obtained from an activity list by decoding it through a serial SGS, and then ordering the activities according to the obtained start times to satisfy the conditions of the TO representation. If TO(S) is the TO representation of a schedule *S*, then *S*(TO(*S*)) = *S*. In other words, this guarantees that any TO representation is uniquely associated with a schedule. We adopt a modified version of the TO representation, explained in the next section.

5.3.2 Original and reversed network

In order to explore a different solution space for the same problem, several authors such as Li and Willis (1992), Özdamar and Ulusoy (1996) and Klein (2000) proposed to reverse the project network and scheduling in backward direction. This is closely related to the concept of FBI discussed in section 5.3.8. Ranjbar et al. (2009) proposed a SS that alternates at each iteration between original and reversed project networks in order to explore a different solution space. The original project network is defined as the network in which the arrows represent the original precedence relations. The reversed project network is obtained by reversing the directions of all the arrows such that the end (start) activity becomes the start (end) activity and each precedence relation $i \rightarrow j$ becomes $j \rightarrow i$. Any feasible schedule for the original (reversed) network is called a forward (backward) schedule. As they obtained promising results, the same mechanism is used in the proposed SS.

In order to embed the TO representation into the reversing of the project network, Ranjbar et al. (2009) adapted the TO representation of Valls et al. (2003) to generate backward (forward) children from forward (backward) solutions. A forward (backward) activity list is first decoded with the serial SGS to obtain the finish times of the activities based on the original (reversed) project network. The activity list is then reordered based on the non-increasing order of their finish times: $f_i > f_j$ implies $p_i < p_j$, while $f_i = f_j$ and $i > j$ implies $p_i < p_j$. The obtained topological order activity list (TOAL) representation is precedence-feasible for the reversed (original) network.

5.3.3 Initial population generation

Regret-biased random sampling (RBRS) with the minimum latest finish time (LFT) is used to generate an initial population of size *InitPop* with the direct project network. Previous studies showed the superiority of random sampling over deterministic approaches (Kolisch, 1996).

Random sampling generates schedules by biasing a priority rule through a random device. Among these methods, RBRS with LFT is one of the most powerful (Kolisch, 1996; Tormos & Lova, 2001; Valls, Ballestin, & Quintanilla, 2005). Each activity list is iteratively created by randomly selecting an activity of the decision set D_k according to the following probability to be placed at position k:

$$\psi(i) = \frac{(\rho_i + 1)}{\sum_{j \in D_k} (\rho_j + 1)}, \text{ with } \rho_i = \max_{j \in D_k} (LF_j) - LF_i, \ \forall i \in D_k \quad (1)$$

where LF_i denotes the latest finish time of activity i calculated by the Critical Path Method (CPM), and the decision set D_k represents the eligible set at stage k. The serial SGS is then applied to decode the generated activity lists into schedules and to evaluate their makespan.

5.3.4 Reverse the type of schedule

Once a new population have been generated, the direction of the project network and the associated type of schedule are reversed from original (reversed) to reversed (original) at the beginning of a new SS iteration. As each forward (backward) activity list of the population has been decoded with the serial SGS, the finish times are used to obtain the TO representation that can be used for scheduling with the reversed (original) network (see section 5.3.2).

5.3.5 Reference set update method

The SS algorithm builds and maintains a subset *RefSet* of high-quality and diverse solutions found during the search. According to the SS terminology proposed in Marti et al. (2006), we use the so-called static Refset Update mechanism, which is detailed in Figure 5.2. It is applied to the new population *New_Pop* during the process once all the subsets have been combined. The reference set *RefSet* is divided into two disjoint subsets $RefSet_1$ and $RefSet_2$ of size b_1 and b_2,

3. Reference set update

a. Sort *New_Pop* according to the lowest makespan and the tie-breaking rule presented in eq. (3).

b. Construction of $RefSet_1$

i. Initialize $RefSet_1$ with the best current solution.

ii. The sorted solutions of *New_Pop* are scanned and added to $RefSet_1$ if the minimum distance with the current members of $RefSet_1$ is superior or equal to t_1.

iii. Improvement Method

1. Set *improve* = 1.

2. While *improve*= 1

a. Identify in *New_Pop* \ $RefSet_1$ the candidates that can replace a member of $RefSet_1$ so that the average makespan of $RefSet_1$ is improved without violating the constraint on the minimum distance or the average distance between the members of $RefSet_1$ is improved without decreasing the average makespan.

b. If the set of candidates is empty, then set *improve* = 0.

c. If the length of *RefSet1* is inferior to b_1, then store the new associated $RefSet_1$ for each candidate and try to expand it with the procedure in 3.b.ii.

d. Select the candidate with the longest associated $RefSet_1$. In case of tie, select the candidate such that the associated $RefSet_1$ has the minimum average makespan. In case of tie select the candidate such that the associated $RefSet_1$ has the maximum average distance between its members.

e. Update $RefSet_1$.

End While

c. Construction of *RefSet2*

i. Initialize *RefSet2* as an empty set.

ii. The sorted solutions of *New_Pop* \ $RefSet_1$ are scanned and added to $RefSet_2$ if the minimum distance with the current members of $RefSet_1 \cup RefSet_2$ is superior or equal to t_2.

iii. Improvement Method: An improvement method similar to 3.b.iii. is used, except that we compute the average distance with any member of $RefSet_1$ and $RefSet_2$.

d. If Incomplete *RefSet* then:

i. If *init* = 1, then:

1. Complete $RefSet_1$ and *RefSet2* by applying the procedures 3.b.ii. and 3.c.ii. without the distance condition.

2. Set *init* = 0;

ii. Else if *init* = 0, then apply the frequency-based diversification strategy.

e. *RefSet* = $RefSet_1 \cup RefSet_2$.

Figure 5.2: Reference set update method

respectively. Each solution of $RefSet_1$ should have a distance of at least t_1 with other solutions of $RefSet_1$, while each solution of $RefSet_2$ should have a distance of at least t_2 (with $t_2 > t_1$) with any solution of $RefSet_1$ or $RefSet_2$.

The distance measure between two solutions represented by AL_1 and AL_2 is the same as in Debels et al. (2006) and Ranjbar and Kianfar (2009) and is defined as follows:

$$\text{Distance}\left(AL_1, AL_2\right) = \frac{1}{n} \cdot \sum_{i=1}^{i=n} \left| p_{i1} - p_{i2} \right| \tag{2}$$

where p_{i1} and p_{i2} are the position of activity i in the activity lists AL_1 and AL_2, respectively. The construction of $RefSet_1$ and $RefSet_2$ starts by sorting the new population *New_Pop* according to the lowest makespan. The following function is considered in case of tie:

$$f(s) = \sum_{i=1}^{i=n} \left(d_i \cdot \frac{f_i - EF_i}{EF_i + 1} \cdot \sum_{k=1}^{K} r_{ik} \right) \text{ for forward schedules,}$$
$$f(s) = \sum_{i=1}^{i=n} \left(d_i \cdot \frac{f_i - \left(LS_n - LS_i\right)}{LS_n - LS_i + 1} \cdot \sum_{k=1}^{K} r_{ik} \right) \text{ for backward schedules} \tag{3}$$

Introduced by Paraskevopoulos et al. (2012), this function depicts for each activity i the deviation of the finish time (f_i) from the earliest finish time (EF_i) and the latest start time (LS_i) calculated by the CPM. This expression is weighted with the activity duration and the sum of the resource consumption. Assuming that high quality solutions have the least deviations from the earliest finish times (latest finish times) in original (reversed) schedules, it can be expected that the lower this function, the better the quality of the solution.

$RefSet_1$ is initialized with the best solution. The next best solutions of *New_Pop* are scanned and added to $RefSet_1$ if the minimum distance with the current members of $RefSet_1$ is superior or equal to t_1. This procedure is similar to the other procedures proposed in the literature that involve distance thresholds (Baradaran, Fatemi Ghomi, Ranjbar, & Hashemin, 2012; Debels et al., 2006; Ranjbar, 2008; Ranjbar et al., 2009; Ranjbar & Kianfar, 2009). However, the problem of building $RefSet_1$ is not trivial and the procedure described above is naïve.

This problem is related to the maximum diversity or dispersion problem introduced by Glover, Hersh G. and McMillian (1977), where the objective is to select a subset of *m* elements from a set of *n* elements in such a way that the distances between the chosen elements are maximized. The problem of building $RefSet_1$ can be formulated as a capacited facility dispersion problem, which is a variant of the problem (Rosenkrantz, Tayi, & Ravi, 2000). In particular, it corresponds to the problem of minimizing the total cost under the constraint that the minimum distance between the facilities should exceeds a threshold. This problem was shown to be NP-hard (Rosenkrantz et al., 2000). For this reason, we propose a new improvement method in a second phase that checks if any solutions of $New\_Pop \setminus RefSet_1$ can be added or can replace a member of $RefSet_1$ to increase the size of $RefSet_1$ (if inferior to b_1), or to improve the quality (average makespan) or the diversity (average distance with the other members of the set) of $RefSet_1$.

$RefSet_2$ is similarly built from the remaining solutions $New\_Pop \setminus RefSet_1$. Depending on the values of b_1, b_2, t_1 and t_2, there may not be enough qualified solutions to complete $RefSet_1$ and $RefSet_2$. If this situation arises when the RefSet Update mechanism is applied to the initial population, we complete the refsets with the best remaining solutions without checking the threshold conditions. Otherwise, a diversification strategy involving a frequency-based memory is used to complete the refsets with new diversified activity lists. We maintain for this purpose a matrix *M* during the search, where each element m_{ij} tracks the number of time activity *i* has been placed directly before activity *j* in all previously generated solutions. A new activity list is built step by step by adding at each stage *k* the activity *j* of the decision set D_k that has the least been placed directly after the activity $a_{k\text{-}1}$ at position (k - 1) in the list under construction: $m_{a_{k-1}\,j} = \min_{r \in D_k} \left(m_{a_{k-1}\,r} \right)$.

5.3.6 Generate the subsets from *Refset*

The subsets generation block forms the subsets of solutions that will be used as basis for creating the combined solutions. In its typical form, the combination method of a SS consists in combining pairs of elements of $RefSet_1$ and $RefSet_2$ (Marti et al., 2006). This method is applied to generate all pairs of solutions in $RefSet_1$, and all combination of one solution from $RefSet_1$ and

another one from $RefSet_2$. Therefore, the total number of pairs to be combined is $\frac{b_1 \cdot (b_1 - 1)}{2} + b_1 \cdot b_2$.

5.3.7 Combination method - Path Relinking

The combination method used to generate the next new population *New_Pop* from the subsets of solutions is presented in Figure 5.3. This evolutionary method was originally designed in the context of tabu search to integrate intensification and diversification strategies, and was later suggested as combination method for SS (Marti et al., 2006; Resende et al., 2010). PR generates new solutions by exploring trajectories that connect high-quality solutions, starting from one solution (the initiating solution) and building a path in the neighborhood space that leads towards other solutions (the guiding solutions). This is accomplished by gradual moves that introduce attributes contained in the guiding solutions.

In our SS algorithm, PR is applied to each pair, which constitutes the initiating and guiding solutions. The moves are performed on the activity list representation of the solutions, AL_1 and AL_2. The proposed PR is a bidirectional PR that swaps the initiating and guiding solutions at each step of the path construction. This advanced type of PR was identified as a promising approach by Glover et al. (1995), Marti et al. (2006) and Resende et al. (2010). Second, we propose a new move that consists in moving the most distant activity from the guiding solution in the current solution of the path to its place in the guiding solution.

We initialize the PR by setting the initiating and guiding solutions with $AL_{guid} = AL_1$, and $AL_{init} = AL_2$, such that the makespan of AL_1 is better than the makespan of AL_2. We consider $AL_{guid} = (a_{0,guid}, \ldots, a_{n+1,guid})$ and $AL_{init} = (a_{0,init}, \ldots, a_{n+1,init})$, where $a_{i,guid}$ and $a_{i,init}$ represent the activity at position i in the activity lists AL_{guid} and AL_{init}, respectively. We denote by *PR_set* the set of activity lists that will be created during the PR process. We introduce the following distance measure for each activity:

$$\text{Distance}(j) = \left| p_{j,init} - p_{j,guid} \right|, \ \forall j \in [1,..,n] \tag{4}$$

where $p_{j,init}$ and $p_{j,guid}$ represent the position of activity j in the initiating and guiding activity lists,

6. PR-based combination method
   a. Let AL_1 be the best *AL* of *Subset* and AL_2 be the remaining *AL*.
   b. Set $AL_{guid} = AL_1$ and $AL_{init} = AL_2$.
   c. Initialize *PR_set* as an empty set.
   d. While Distance(AL_{init}, AL_{guid}) > 0 do
      i. Determine the most distant activity *q* between AL_{init} and AL_{guid}.
      ii. Set $AL_{cu} = AL_{init}$.
      iii. Move *q* from position p_{init} to position p_{guid} in AL_{cu}.
      iv. If AL_{cu} is not precedence feasible, then use the repair mechanism.
      v. Set $AL_{init} = AL_{guid}$ and $AL_{guid} = AL_{cu}$.
      vi. If Distance(AL_{init}, AL_{guid}) > 0, Add AL_{cu} to *PR_set*.
      End While
   e. Divide *PR_set* into n_{pr} subsets of size $|PR\_set| / n_{pr}$.
   f. For i =1 to n_{pr} do
      i. Randomly select a single list *AL* from the i^{th} subset of *PR_set*.
      ii. Schedule *AL* with the serial SGS (if *rev* = 0, forward scheduling, else backward scheduling).
      iii. If the makespan associated with *AL* is inferior or equal to the best current makespan, then invoke the improvement strategy based on FBI.
      iv. Add *AL* to the *New_Pop*.
      End For

Figure 5.3: PR-based combination method

respectively. Let *q* be the most distant activity between AL_{init} and AL_{guid}, p_{init} be the position of *q* in AL_{init} ($a_{p_{init},init} = q$), and p_{guid} be the position of *q* in AL_{guid} ($a_{p_{guid},guid} = q$). In case of tie, the activity with the lowest index is selected among the most distant activities. From AL_{init}, the current activity list AL_{cu} is built by moving activity *q* at position p_{guid} as follows:

1) If pinit < pguid, then

$$AL_{cu} = \left(a_{0,init},...,a_{(p_{init}-1),init}, a_{(p_{init}+1),init},...,a_{p_{guid},init}, q, a_{(p_{guid}+1),init},...,a_{n+1,init}\right);$$

2) If pinit > pguid, then

$$AL_{cu} = \left(a_{0,init},...,a_{(p_{guid}-1),init}, q, a_{p_{guid},init},...,a_{(p_{init}-1),init}, a_{(p_{init}+1),init},...,a_{n+1,init}\right).$$

If the current activity list is not precedence-feasible, the following repair mechanism is used:

1) If $p_{init} < p_{guid}$, then for forward (backward) schedules all successors (predecessors) of q between positions p_{init} and p_{guid} - 1 in AL_{cu} are moved right after q in the same order;

2) If $p_{init} > p_{guid}$, then for forward (backward) schedules all predecessors (successors) of q between position p_{guid} +1 and p_{init} in AL_{cu} are moved right before q in the same order.

The resulting AL_{cu} is added to *PR_set*. We then set $AL_{init} = AL_{guid}$ and $AL_{guid} = AL_{cu}$ and repeat the process until $AL_{cu} = AL_{guid}$, thus connecting AL_1 and AL_2 with a single path composed of all the activity lists stored in *PR_set*. Only a subset of *PR_set* will be selected for evaluation with the serial SGS. Let n_{pr} be the number of activity lists to be selected for evaluation. The activity lists in *PR_set* are numbered from 1 to |*PR_set*| and placed in n_{pr} subsets of size |*PR_set*| / n_{pr}. One AL from each subset is randomly selected and evaluated to find its schedule and its makespan.

We illustrate the combination method with an example project depicted in Figure 5.4. We assume that AL_1 = (1,3,2,5,4,7,8,6,11,9,10,12) and AL_2 = (1,2,3,4,6,5,8,9,7,10,11,12). The combination method is presented in Table 5.1. In the first step, we set $AL_{guid} = AL_1$, and $AL_{init} = AL_2$. The most distant activity is q = 6 and its position in the initial and guiding lists are p_{init} = 5 and p_{guid} = 8, respectively. From AL_{init}, AL_{cu} is built by moving activity 6 at position 8. However, this activity list is not precedence-feasible because one of its successors, activity 9, is placed before activity 6 in the list. The repair mechanism is thus invoked in order to obtain a precedence-feasible activity list. We then set $AL_{init} = AL_{guid}$ and $AL_{guid} = AL_{cu}$ and the process is repeated until $AL_{cu} = AL_{guid}$ in step 7. The 6 precedence-feasible activity lists created during the process are stored in *PR_set*. If we assume that n_{pr} = 1, a single activity list is randomly selected from PR_set to be evaluated.

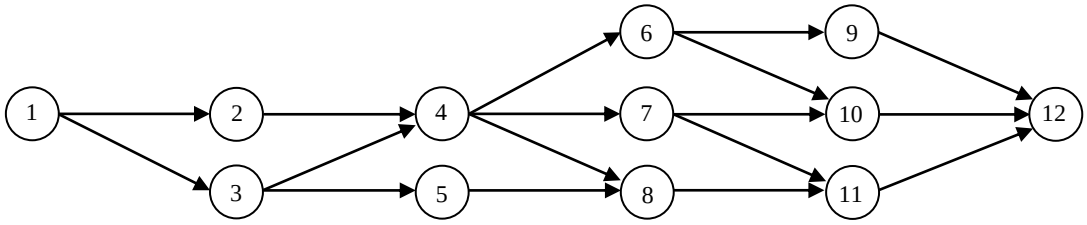

Figure 5.4: Example project

Table 5.1: Illustration of the combination method

AL_1: 1 3 2 5 4 7 8 6 11 9 10 12 $\quad$ AL_2: 1 2 3 4 6 5 8 9 7 10 11 12

Step	AL_{init}	$ALguid$	q	p_{init}	p_{guid}	AL_{cu}	feasibility
1	1 2 3 4 6 5 8 9 7 10 11 12	1 3 2 5 4 7 8 6 11 9 10 12	6	5	8	1 2 3 4 5 8 9 6 7 10 11 12	Infeasible
repair						1 2 3 4 5 8 6 9 7 10 11 12	Feasible
2	1 3 2 5 4 7 8 6 11 9 10 12	1 2 3 4 5 8 6 9 7 10 11 12	7	6	9	1 3 2 5 4 8 6 11 7 9 10 12	Infeasible
repair						1 3 2 5 4 8 6 7 11 9 10 12	Feasible
3	1 2 3 4 5 8 6 9 7 10 11 12	1 3 2 5 4 8 6 7 11 9 10 12	9	8	10	1 2 3 4 5 8 6 7 10 9 11 12	Feasible
4	1 3 2 5 4 8 6 7 11 9 10 12	1 2 3 4 5 8 6 7 10 9 11 12	10	11	9	1 3 2 5 4 8 6 7 10 11 9 12	Feasible
5	1 2 3 4 5 8 6 7 10 9 11 12	1 3 2 5 4 8 6 7 10 11 9 12	2	2	3	1 3 2 4 5 8 6 7 10 9 11 12	Feasible
6	1 3 2 5 4 8 6 7 10 11 9 12	1 3 2 4 5 8 6 7 10 9 11 12	4	5	4	1 3 2 4 5 8 6 7 10 11 9 12	Feasible
7	1 3 2 4 5 8 6 7 10 9 11 12	1 3 2 4 5 8 6 7 10 11 9 12	9	10	11	1 3 2 4 5 8 6 7 10 11 9 12	Feasible

5.3.8 Improvement strategy - FBI

Search intensification is typically achieved in SS with the execution of an improvement method applied to the new solutions found during the combination process. As pointed by Marti et al. (2006), an important issue in SS design is how to allocate the computational effort between improving current solutions and generating new ones. In the existing literature on SS applied to RCPSP, authors have given various levels of priority to improvement. At the one extreme, Ranjbar et al. (2009) did not apply any improvement method. At the other extreme, Paraskevopoulos et al. (2012) introduced a local search and a perturbation strategy based on a long-term memory, while Valls et al. (2004) and Chen et al. (2010) applied a procedure that improves the local resource utilization until no further improvement can be produced. In between, Debels et al. (2006) and Mobini et al. (2009) applied the well-known FBI procedure. Introduced by Li and Willis (1992) and popularized by Tormos and Lova (2001) and Valls et al. (2005), FBI iteratively applies forward and backward passes until no further improvement can be produced. In the forward (backward) pass, the activities are scheduled as early as possible (as late

as possible). As this simple technique produces notable improvements in schedules quality with a small computation effort (Valls et al., 2005), it is introduced in the proposed SS.

In the first phase of the FBI, the backward pass is done by applying the transformation from a AL to a TOAL to obtain a backward (forward) TOAL from a forward (backward) AL, reversing the project network from original (reversed) to a reversed (original) network, and then scheduling the TOAL with the reversed (original) network. In the second phase, the forward pass is applied in the same manner to obtain a forward (backward) schedule in the initial original (reversed) network direction. As proved in Valls et al. (2005), the makespan of the schedule after an iteration of FBI is inferior or equal to the initial makespan. This process is repeated if the makespan has been improved. In the proposed SS, FBI is first applied to the best solutions of the initial population, and then only on high-quality solutions that match or improve the best makespan found so far during the process.

5.3.9 New Population generation

The new population *New_Pop* is first initialized with the best solution found so far. In case of tie, the rule proposed in section 5.3.5 is considered. For each subset, n_{pr} activity lists are generated with PR, scheduled with the serial SGS and possibly improved by FBI. This constitutes the new population for the next SS iteration, thus composed of $\left(\frac{b_1 \cdot (b_1 - 1)}{2} + b_1 \cdot b_2\right) \cdot n_{pr} + 1$ solutions.

5.4 Computational analysis

This section presents an evaluation of the proposed SS on the PSPLIB data sets proposed in Kolisch and Sprecher (1997), and a comparison with the state-of-the-art heuristics developed for the RCPSP. The input parameters of the SS algorithm are first determined empirically by parameter tuning. The experiments were conducted on a personal computer with an Intel Core I5 2.53 GHz processor and 4 GB RAM under Windows 7 Professional. The algorithm was implemented in Matlab R2011b.

5.4.1 PSPLIB data sets and test design

As test instances, we use the standard J30, J60 and J120 sets of the PSPLIB library, which have been generated using PROGEN (Kolisch, Sprecher, & Drexl, 1992; Kolisch et al., 1995). The J30 and J60 sets consist of 480 projects of 30 and 60 activities, respectively. The J120 set is composed of 600 projects of 120 activities. For more details on how the projects data are generated, the reader is referred to Kolisch et al. (1992) and Kolisch et al. (1995). The complete data are available from the project scheduling library PSPLIB on the internet. Even though other benchmark instances exist in the literature, such as the Patterson instances (Patterson, 1984), the PSPLIB library is the most widely used for experimental comparison of heuristics for the RCPSP.

An experimental protocol for testing the heuristics with the PSPLIB instances has been proposed by Hartmann and Kolisch (2000) and Kolisch and Hartmann (2006). Three stopping criteria are considered: 1000, 5000 and 50000 generated schedules. Since the speed of computers has increased and since some researchers have observed significant improvement in the results with larger schedule limits, our method is also tested with 500000 schedules. A schedule-based stopping criterion is used because the corresponding computational effort is independent from the evolution of the speed of computers, the operating systems, the compilers and the implementational skills and is then quite similar for all the tested heuristics (Kolisch & Hartmann, 2006). The heuristics are then compared according to the average deviation from the optimal solutions (for J30) or from the critical path-based lower bound (for J60 and J120).

5.4.2 Impact of randomness on the performance

As for most metaheuristic approaches used to solve the RCPSP, many parts of the proposed SS are based on random devices. In our algorithm, randomness is involved in the initial population generation, the selection of the n_{pr} activity lists from *PR_set*, and the diversification to complete $RefSet_1$ and $RefSet_2$ when required. Each execution for the same problem instance would provide different results when random devices are used. However, most papers on metaheuristic approaches for the RCPSP do not indicate if multiple executions are conducted. In Hartmann and Kolisch (2000), Kolisch and Hartmann (2006) and in the papers published afterwards, the performance measure used for sorting the metaheuristics is the average deviation measured with a

single figure in percentage with a two-digit precision. However, the value of the schedule limits were arbitrarily set in order to provide a unified base of comparison of the performances of heuristics, but they do not guarantee a convergence of the results and a margin of errors due to randomness in accordance with a two-digit precision. In order to take into account the effect of randomness in our experiments, each instance is solved ten times and the average results of the ten replications, as well as the 95% confidence interval, are provided.

5.4.3 Parameter tuning

The proposed SS is based on six input parameters: *InitPop*, b_1, b_2, t_1, t_2 and n_{pr}. Preliminary tests showed that the performances of the SS are highly sensitive to the values of these parameters. Thus, their values have to be carefully set with parameter tuning. Among the methods used for varying the parameters in the literature, most authors used full factorial or Taguchi designs of experiment, or elementary trial and error strategies. Tuning is usually performed on the whole standard sets, or a subset of instances or a new sample of instances. Finally, parameter tuning can be conducted for each instance size, for each schedule limit or each combination of schedule limit and project size. Based on preliminary tests, we noted that the promising combinations had different parameters values depending on the schedule limit and the project size. Therefore, parameter tuning was independently performed on each combination of schedule limit and project size. We also observed that a full factorial design with six parameters would not be efficient to explore the space of combinations and would be time consuming, unless the subsets used for tuning are very small or unless the number of repetitions is reduced. However, a weak correlation was observed between the performances for very small subsets of instances or for few repetitions and the performances on the whole standard sets with ten repetitions.

As an alternative method, Talbi (2009) suggested that the identification of appropriate values for the parameters can be formulated as an optimization problem and solved by another metaheuristic, leading to a meta-algorithm composed of two levels. At the meta-level, a metaheuristic operates on solutions representing the parameters of the metaheuristic to optimize. The metaheuristic at the base level is used to solve the original optimization problem. Following this framework, we propose a basic local search process for the meta-level which moves from a combination of parameters to another. Starting from an initial combination of parameters, each parameter is changed to a higher or a lower value with a predefined step, one at a time. The

combination with the lowest average deviation in this neighborhood is used as starting combination for the next iteration. To avoid being trapped in local optima, we authorize non-improving iterations. Visited combinations are memorized to avoid cycling and to restart the process from the best not selected combinations after a number of non-improving iterations. The local search is stopped after a number of non-improving iterations, sometimes after several hundreds of tested combinations.

This method is applied to find the best combinations of parameters for each standard set (J30, J60, J120) and each schedule limit (1000, 5000, 50000, 500000 schedules). The local search is applied to the whole standard sets and ten independent runs are conducted, except for the J60 set with 500000 schedules and for the J120 set with 50000 and 500000 schedules. For these combinations, the number of instances and the number of independent runs are reduced. For instance, only five independent runs are conducted on a subset of the hardest instances for J120 with 50000 schedules (i.e., the instances with a deviation higher than 4% between the makespan obtained with 5000 schedules and the best known makespan in literature). Table 5.2 presents the resulting combinations of parameters for each instance set and each schedule limit. No claim is made that these parameter values are the best possible. It is likely that a different parameter tuning method would have produced different results.

To analyze the values obtained with parameter tuning, we first focus on the qualitative influence of the parameters on the behavior of the scatter search. A small value of *InitPop* will allow more effort to be put into the subsequent scatter search mechanisms, which are more efficient for finding good solutions than the random generation used in the initial population generation. At the opposite, more diversified local optimums can be found with a large value of *InitPop*. The parameters b_1, b_2 and n_{pr} directly influence the size of the population tested at each iteration and thus the number of iterations for a given schedule limit. Finally, b_1, b_2, t_1 and t_2 have an influence on the quality and the diversity of the reference set used for combination.

A general analysis of the combinations of parameters shows that $n_{pr} = 1$ gives the best results and is not sensitive to the other parameters. In addition, the most influent parameters seem to be b_1 and b_2. The parameters t_1 and t_2 also seem to have an important effect on the performances of the scatter search. *InitPop* has less impact on the results, especially for large schedule limits. Several

Table 5.2: Parameter tuning

Instance set	**J30**			**J60**				**J120**			
Schedule limit	**1000**	**5000**	**50000**	**1000**	**5000**	**50000**	**500000**	**1000**	**5000**	**50000**	**500000**
InitPop	100	100	1000	100	200	1500	7500	100	100	1100	3000
b_1	8	15	35	7	10	27	75	5	9	17	40
b_2	4	16	29	4	9	21	65	4	7	17	40
t_1	0.8	0.6	0.5	0.8	0.9	1.3	1.2	0.8	1.7	2.2	2.3
t_2	1.5	1.5	1.4	1.6	1.8	2.1	2.3	1.9	2.7	3.8	3.7
n_{pr}	1	1	1	1	1	1	1	1	1	1	1

trends emerge from the combinations in Table 5.2. The value of *InitPop* is positively related to the schedule limit. Both b_1 and b_2 decrease when the schedule limit decreases and when the size of the projects increases so that more iterations of the scatter search are performed. Both t_1 and t_2 increase when the schedule limit increases. This may reflects that more diversity is required for large schedule limits in order to explore more thoroughly the solution space.

5.4.4 Detailed results of the scatter search algorithm

The detailed experimental results are presented in Table 5.3. The row labeled “Sum” gives the average sum of the makespans of all the instances of the set with ten independent runs. The row labeled “Avg. dev. CPM”, “Avg. dev. LB” and “Avg. dev. best” indicate the average deviation from the critical path lower bound, the best known lower bound and the current best known makespan, respectively. For the latter value, the reader is referred to the PSPLIB website which indicates for each instance the best known makespan and the author who found the solution. The results available on January 1, 2014 were used. “No. of instances” represents the number of instances in the set, while “Av. No. of best” is the number of instances for which the makespan obtained with the proposed method is as good as the current best known makespan. “Avg. CPU” and “Max. CPU” indicate the average and maximum computation times to reach the best solution, respectively. “Avg. No. sched.” represents the average number of generated schedules to reach the best solution.

Table 5.3 highlights that the algorithm is able to find all the optimal solutions of J30 with 50000 schedules in reasonable computation times. Near-optimal solutions can even be found with 1000 and 5000 schedules for J30 in very small computation times. For J60, our method provides an

Table 5.3: Detailed experimental results for ten repetitions

Instance set	**J30**			**J60**				**J120**			
Schedule limit	**1000**	**5000**	**50000**	**1000**	**5000**	**50000**	**500000**	**1000**	**5000**	**50000**	**500000**
Sum	28354	28324	28316	38647	38495	38374	38333	76125	75205	74437	74001
Avg. dev. CPM	13.53%	13.40%	13.37%	11.38%	10.93%	10.58%	10.45%	34.13%	32.52%	31.16%	30.39%
Avg. dev. LB	0.10%	0.02%	0.00%	1.33%	1.02%	0.77%	0.68%	6.27%	5.18%	4.29%	3.78%
Avg. dev. best	0.10%	0.02%	0.00%	0.68%	0.38%	0.13%	0.05%	3.09%	2.06%	1.22%	0.74%
No. of instances	480	480	480	480	480	480	480	600	600	600	600
Avg. No. of best	452	474	480	371	392	427	454	206	230	260	301
Avg. CPU (s)	0.27	0.80	3.50	0.90	3.24	34.66	735.60	4.62	21.04	198.14	2976.53
Max. CPU (s)	2.80	15.19	191.02	5.32	26.42	369.89	12800.24	10.69	83.93	647.67	14179.87
Avg. No. sched.	100	248	760	176	610	4578	26979	457	1917	15699	117526

average deviation of 0.05 % with the best known makespans for 500000 schedules, and reaches the best solutions in 454 instances out of 480. Similarly, we obtain for J120 and 500000 schedules an average deviation of 0.74% with the best known makespans, reaching the best solutions in 301 instances out of 600. This is remarkable as the best known makespans have been obtained with any stopping criterion and any existing heuristic in the literature, and as there is currently no unique method that can reach all the best known solutions for J60 and J120. In addition, the proposed SS is able to find attractive solutions with lower schedule limits. For instance, we obtain an average deviation of only 0.68% with the best known makespans for J60 with 1000 schedules in less than 1 second.

Table 5.4 summarizes the detailed results for each repetition (“Dev. (%) rep.”), as well as the average deviation for ten repetitions (“Avg. Dev. (%)”), the minimum and maximum deviation observed (“Min. Dev. (%)” and (“Max. Dev. (%)”, resp.) and the 95% confidence interval (“95% Conf. int. (%)”). It reveals that the larger the size of the project or the lower the schedule limit,

the wider the 95 % confidence interval. A statistical test is performed on each combination of size and schedule limit to test whether replicating the experiments has a significant impact on the results. Since the deviations are not normally distributed and the variances are not homogeneous, only non parametric tests can be employed. The Friedman test demonstrates that the effect of the replications is not significant in all cases with a five percent confidence level.

Table 5.4: Overview of the average deviation for ten repetitions

Instance set	**J30**			**J60**				**J120**			
Schedule limit	**1000**	**5000**	**50000**	**1000**	**5000**	**50000**	**500000**	**1000**	**5000**	**50000**	**500000**
Dev. (%) rep. 1	0.11	0.02	0.00	11.39	10.94	10.57	10.46	34.16	32.50	31.15	30.37
Dev. (%) rep. 2	0.11	0.02	0.00	11.39	10.93	10.58	10.45	34.25	32.51	31.12	30.37
Dev. (%) rep. 3	0.09	0.03	0.00	11.43	10.95	10.60	10.46	34.15	32.54	31.15	30.36
Dev. (%) rep. 4	0.09	0.02	0.00	11.43	10.95	10.59	10.46	34.14	32.54	31.16	30.40
Dev. (%) rep. 5	0.11	0.01	0.00	11.34	10.89	10.58	10.45	34.07	32.46	31.19	30.39
Dev. (%) rep. 6	0.10	0.02	0.00	11.34	10.89	10.57	10.45	34.13	32.53	31.16	30.44
Dev. (%) rep. 7	0.10	0.01	0.00	11.36	10.96	10.57	10.45	34.14	32.51	31.20	30.39
Dev. (%) rep. 8	0.10	0.02	0.00	11.44	10.96	10.56	10.46	34.09	32.48	31.14	30.44
Dev. (%) rep. 9	0.11	0.03	0.00	11.36	10.92	10.56	10.45	34.03	32.56	31.17	30.39
Dev. (%) rep. 10	0.12	0.02	0.00	11.38	10.95	10.58	10.44	34.19	32.54	31.19	30.40
Avg. Dev. (%)	0.10	0.02	0.00	11.38	10.93	10.58	10.45	34.13	32.52	31.16	30.39
Min. Dev. (%)	0.09	0.01	0.00	11.34	10.89	10.56	10.44	34.03	32.46	31.12	30.36
Max. Dev. (%)	0.12	0.03	0.00	11.44	10.96	10.60	10.46	34.25	32.56	31.20	30.44
95% Conf. int. (%)	[0.10; 0.11]	[0.02; 0.02]	[0.00; 0.00]	[11.36; 11.41]	[10.92; 10.95]	[10.57; 10.58]	[10.45; 10.46]	[34.10; 34;17]	[32.50; 32.54]	[31.15; 31.18]	[30.38; 30.41]
Width of the 95% Conf. int. (%)	0.01	0.01	0.00	0.05	0.03	0.02	0.01	0.08	0.04	0.03	0.04

5.4.5 Comparative analysis with different versions of the algorithm

The main novelties of the proposed algorithm are the bidirectional PR based on a new move and the new improvement method used in the Refset update mechanism. In order to prove their efficiency, three additional versions of the algorithm were developed and compared to the original algorithm. Two versions of the algorithm called SS_PR1 and SS_PR2 differ from the original version only in the PR. SS_PR1 is based on the PR used in Mobini et al. (2009) and Baradaran, Fatemi Ghomi, Mobini and Hashemin (2010), while SS_PR2 is based on the PR used in Ranjbar et al. (2009) and Baradaran et al. (2012). The third version of the algorithm called SS_RefSet is created by removing the improvement method used in the Refset update mechanism from the algorithm.

Table 5.5 shows the comparative results of the original algorithm and the three additional versions of the algorithm. It highlights that the best average deviations are always obtained with the original algorithm. However, we cannot exclude that the difference between the original algorithm and the other versions of the algorithm is significant for certain combinations of size and schedule limit because the corresponding 95% confidence intervals overlap. Statistical tests are further required to assess if the differences are statistically significant. The Friedman test reveals that the repetitions do not have a significant effect on the performances with a five percent confidence level. The Wilcoxon test is used to compare each alternative version of the algorithm with the original algorithm, for each combination of size and schedule limit. As shown in Table 5.5, the performances of the original algorithm and the alternative versions of the algorithm are significantly different with a five percent confidence level in most cases. In addition, the computation times for the three alternative versions of the algorithm are very close to the computation times observed for the original algorithm.

It confirms that the new PR mechanism is significantly better than other PR used in the literature and that the improvement algorithm used in the Refset update mechanism also provides a significant improvement in the performances of the proposed algorithm. These results are obtained without increasing the computation times.

Table 5.5: Comparison of different versions of the Scatter Search algorithm

Instance set	**J30**			**J60**			**J120**		
Schedule limit	**1000**	**5000**	**50000**	**1000**	**5000**	**50000**	**1000**	**5000**	**50000**
SS	0.10[a] [0.10;0.11][b]	0.02 [0.02;0.02]	0.00 [0.00;0.00]	11.38 [11.36;11.41]	10.93 [10.92;10.95]	10.58 [10.57;10.58]	34.13 [34.10;34.17]	32.52 [32.50;32.54]	31.16 [31.15;31.18]
SS_PR1	0.15 [0.14;0.16]	0.05 [0.05;0.06]	0.01 [0.00;0.01]	11.54 [11.53;11.55]	11.02 [11.01;11.04]	10.65 [10.64;10.66]	34.49 [34.44;34.55]	32.74 [32.71;32.76]	31.18 [31.15;31.21]
SS_PR2	0.13 [0.12;0.14]	0.03 [0.03;0.04]	0.00* [0.00;0.00]	11.51 [11.49;11.52]	10.96* [10.94;10.98]	10.61 [10.60;10.62]	34.37 [34.35;34.39]	32.62 [32.60;32.64]	31.17* [31.15;31.20]
SS_RefSet	0.11* [0.11;0.12]	0.03 [0.03;0.03]	0.01 [0.00;0.01]	11.42 [11.40;11.44]	10.97 [10.96;10.98]	10.61 [10.60;10.63]	34.21 [34.15;34.26]	32.56* [32.54;32.59]	31.21 [31.20;31.22]

[a] Average deviation from the optimal solutions (for J30) or from the critical path-based lower bound (for J60 and J120) for ten repetitions.
[b] 95% confidence interval of the average deviation with ten repetitions.
* Non-significant difference at the five percent level of confidence.

5.4.6 Comparative analysis with the best metaheuristic approaches

The comparative analysis is divided into two sections. In the first section, the results of the proposed approach are compared with the best heuristics that provided results following the experimental protocol summarized in section 5.4.1. In the second section, we compare the results of the heuristics that do not use any schedule limit.

5.4.6.1 Comparative analysis with schedule limits

For reasons of space, the comparison is limited to the twenty best heuristics with 50000 schedules for either J30, J60 or J120. From the state-of-the-art heuristics surveyed in Kolisch and Hartmann (2006), only the heuristics proposed by Kochetov and Stolyar (2003), Alcaraz, Maroto and Ruiz (2004), Valls et al. (2005) and Debels et al. (2006), which were all identified as the best heuristics at this time, are still currently in this list. This highlights how fruitful the recent developments in heuristics for the RCPSP are. The computational results of the selected heuristics can be found in Tables 5.6, 5.7 and 5.8 for the instance sets J30, J60, and J120, respectively. All these methods are metaheuristics. As explained before, most of these metaheuristics are stochastic, leading to different results at each execution. However, most authors did not indicate if their results were obtained with a single run, or with several runs, and, in the latter case, if their results were the best or the average among several runs. In order to compare our results, the average deviation observed with ten independent runs of the proposed SS algorithm is considered.

Table 5.6: Computational comparison with schedule limits - J30

Algorithm	SGS	Authors	Average Deviation (%)			>50000	
			1000	5000	50000	Avg. dev. (%)	No. of sched.
Best known makespans	**—**	**—**	**0**				
SAILS	Serial	Paraskevopoulos et al. (2012)	0.03	0.01	0		
SS—FBI	**Serial**	**This study**	**0.10**	**0.02**	**0**		
Artificial Immune Algo.—FBI	Serial	Mobini, Mobini and Rabbani (2011)	0.05	0.03	0		
SS—FBI	Serial	Ranjbar et al. (2009)	0.10	0.03	0		
GA, TS—PR	Both	Kochetov and Stolyar (2003)	0.10	0.04	0		
GA—FBI	Both	Wang, Li and Lin (2010)	0.14	0.04	0		
GA	Both	Zamani (2013)	0.14	0.04	0		
Bees Algo.	—	Sadeghi, Kalanaki, Noktehdan, Samghabadi and Barzinpour (2011)	0.15	0.09	0		
ESS	Both	Mobini et al. (2009)	0.05	0.02	0.01		
GA	Both	Mendes, Goncalves and Resende (2009)	0.06	0.02	0.01		
GA—FBI	Serial	Goncalves, Resende and Mendes (2011)	0.32	0.02	0.01	0.01	499860
GA—DP	Both	Cervantes, Lova, Tormos and Barber (2008)	0.16	0.04	0.01		
PSO—HH	Serial	Koulinas, Kotsikas and Anagnostopoulos (2014)	0.26	0.04	0.01		
DABC	Serial	Nouri, Krichen and Ladhari (2013)	0.21	0.05	—		
ACOSS	Both	Chen et al. (2010)	0.14	0.06	0.01		
Cooling Process—GA—FBI	Serial	Lim, Ma, Rodrigues, Tan and Xiao (2013)	0.21	0.07	0.01	0.00	500000
SS—FBI	Serial	Debels et al. (2006)	0.27	0.11	0.01	0.01	500000
GA—FBI	Serial	Debels and Vanhoucke (2007)	0.15	0.04	0.02		
GA—hybrid, FBI	Serial	Valls, Ballestin and Quintanilla (2008)	0.27	0.06	0.02		
Memetic Algo. —FBI	Serial	Carlier et al. (2009)	0.32	0.11	0.02		
GA—FBI	Serial	Valls et al. (2005)	0.34	0.20	0.02		
GA—FBI	Both	Alcaraz et al. (2004)	0.25	0.06	0.03		
SFLA—FBI	Serial	Fang and Wang (2012)	0.36	0.21	0.18		
GANS	Serial	Proon and Jin (2011)	1.83	1.27	0.71		

The first column of Tables 5.6, 5.7 and 5.8 briefly describes each heuristic with the abbreviations used by their authors or commonly used in literature: GA for Genetic Algorithm, SAILS for Scatter Search with Adaptive Iterated Local Search, ESS for enhanced Scatter Search, TS for

Tabu Search, PSO-HH for Particle Swarm Optimization based Hyper-Heuristic algorithm, DABC for Discrete Artificial Bee Colony algorithm, ACOSS for Ant Colony Optimization and SS, SFLA for Shuffled Frog-leaping Algorithm, GANS for GA with Neighborhood Search. The first line "Best known makespans" provides the average deviation based on the current best known makespans obtained with any heuristic and any stopping criterion (see section 5.4.4). The second column indicates the type of SGS employed. The average deviations for 1000, 5000 and 50000 schedules appears in the fourth, fifth and sixth columns, respectively. For each instance set, the heuristics are sorted in ascending order according to the average deviation with 50000 schedules. In case of tie, the results for 5000 and then 1000 schedules are considered. The seventh and eighth columns indicate the average deviation and the schedule limit for the heuristics tested with more than 50000 schedules.

As highlighted in Table 5.6, the proposed SS is only slightly outperformed for J30 by the method of Paraskevopoulos et al. (2012). Our approach belongs to the metaheuristics able to reach all the optimal solutions with 50000 schedules. If we consider the best algorithms with 5000 or 50000 schedules, most of them have a difference of less than 0.01 point from each other. Assuming that the width of the 95% confidence interval for any metaheuristic has the same order of magnitude as the one observed in Table 5.4for our SS (0.01 point for 1000 and 5000 schedules, 0.00 point for 50000 schedules), we can conclude that the precision used by researchers is appropriate and that randomness does not significantly change the current ranking.

Table 5.7 shows that none of the heuristics is able to find all the best known solutions for J60. The proposed algorithm is ranked 7th for 50000 schedules based on the average deviation. However, only three algorithms in the literature are better than the lower bound of the 95% confidence interval presented in Table 5.4 (10.57%). The proposed SS is ranked 6th for 1000 and 5000 schedules based on the average deviation, indicating that it is also competitive with lower schedule limits. As for J30, the average deviations obtained by the best metaheuristics for 50000 schedules are also close to each other. However, assuming that the width of 95% confidence interval for any metaheuristic has the same order of magnitude as the one observed in Table 5.4 (0.05, 0.03 and 0.02 point for 1000, 5000 and 50000, resp.), it may be inappropriate to rank the methods with the precision commonly used by researchers. It would be more appropriate to gather them in groups where the difference in performances could be explained by randomness.

Table 5.7: Computational comparison with schedule limits - J60

Algorithm	SGS	Authors	Average Deviation (%)			>50000	
			1000	5000	50000	Avg. dev. (%)	No. of sched.
Best known makespans	**—**	**—**	**10.37**				
GANS	Serial	Proon and Jin (2011)	11.35	10.53	10.52		
SAILS	Serial	Paraskevopoulos et al. (2012)	11.05	10.72	10.54	10.46	70000
Artificial Immune Algo.—FBI	Serial	Mobini et al. (2011)	11.17	10.80	10.55		
ESS	Both	Mobini et al. (2009)	11.12	10.74	10.57		
GA—FBI	Both	Wang et al. (2010)	11.55	10.96	10.57		
GA—FBI	Serial	Goncalves et al. (2011)	—	11.56	10.57	10.49	499140
SS—FBI	**Serial**	**This study**	**11.38**	**10.93**	**10.58**	**10.45**	**500000**
Cooling Process—GA—FBI	Serial	Lim et al. (2013)	11.73	11.14	10.63	10.51	500000
SS—FBI	Serial	Ranjbar et al. (2009)	11.59	11.07	10.64		
GA	Both	Zamani (2013)	11.33	10.94	10.65		
SFLA—FBI	Serial	Fang and Wang (2012)	11.44	10.87	10.66		
ACOSS	Both	Chen et al. (2010)	11.75	10.98	10.67		
GA	Both	Mendes et al. (2009)	11.72	11.04	10.67	10.67	63546
GA	Serial	Debels and Vanhoucke (2007)	11.45	10.95	10.68		
PSO—HH	Serial	Koulinas et al. (2014)	11.74	11.13	10.68		
Memetic Algo. —FBI	Serial	Carlier et al. (2009)	11.62	11.09	10.70		
SS—FBI	Serial	Debels et al. (2006)	11.73	11.10	10.71	10.53	500000
GA—hybrid, FBI	Serial	Valls et al. (2008)	11.56	11.10	10.73		
GA, TS—PR	Both	Kochetov and Stolyar (2003)	11.71	11.17	10.74		
GA—FBI	Serial	Valls et al. (2005)	12.21	11.27	10.74		
Bees Algo.	—	Sadeghi et al. (2011)	11.93	11.48	10.74		
GA—DP	Both	Cervantes et al. (2008)	11.43	10.96	10.81		
DABC	Serial	Nouri et al. (2013)	11.74	11.16	—		
GA—FBI	Both	Alcaraz et al. (2004)	11.89	11.19	10.84		

With this framework, Proon and Jin (2011) developed the best metaheuristic, followed by a group composed of Paraskevopoulos et al. (2012) and Mobini et al. (2011), and another group composed of this paper, Mobini et al. (2009), Wang et al. (2010) and Goncalves et al. (2011). To further illustrate the impact of randomness on the performances of the proposed SS, note that with *InitPop* = 1500, b_1 = 27, b_2 = 21, t_1 =1.1, t_2 = 2.1, n_{pr} = 1, the minimum average deviation among ten runs is 10.52%, which would match the results obtained by Proon and Jin (2011), but the maximal measure is 10.62%, leading to an average value of 10.59%.

Table 5.8: Computational comparison with schedule limits - J120

Algorithm	SGS	Authors	Average Deviation (%)			>50000	
			1000	5000	50000	Avg. dev. (%)	No. of sched.
Best known makespans	**—**	**—**	**29.18**				
GANS	Serial	Proon and Jin (2011)	33.45	31.51	30.45		
ACOSS	Both	Chen et al. (2010)	35.19	32.48	30.56		
Cooling Process—GA—FBI	Serial	Lim et al. (2013)	34.95	32.75	30.66	29.91	500000
SAILS	Serial	Paraskevopoulos et al. (2012)	33.32	32.12	30.78	30.39	200000
GA	Serial	Debels and Vanhoucke (2007)	34.19	32.34	30.82		
SFLA—FBI	Serial	Fang and Wang (2012)	34.83	33.20	31.11		
SS—FBI	**Serial**	**This study**	**34.13**	**32.52**	**31.16**	**30.39**	**500000**
PSO—HH	Serial	Koulinas et al. (2014)	35.20	32.59	31.23		
GA—hybrid, FBI	Serial	Valls et al. (2008)	34.07	32.54	31.24	30.95	100000
GA—FBI	Both	Wang et al. (2010)	35.18	33.11	31.28		
GA	Both	Zamani (2013)	34.02	32.89	31.30		
ESS	Both	Mobini et al. (2009)	34.51	32.61	31.37	31.20	127341
GA	Both	Mendes et al. (2009)	35.87	33.03	31.44		
Artificial Immune Algo.—FBI	Serial	Mobini et al. (2011)	34.01	32.57	31.48		
Memetic Algo. —FBI	Serial	Carlier et al. (2009)	34.89	33.18	—		
SS-FBI	Serial	Ranjbar et al. (2009)	35.08	33.24	31.49		
DABC	Serial	Nouri et al. (2013)	36.40	33.72	31.49		
GA—FBI	Both	Alcaraz et al. (2004)	36.53	33.91	31.49		
Bees Algo.	—	Sadeghi et al. (2011)	35.80	33.33	31.55		
SS—FBI	Serial	Debels et al. (2006)	35.22	33.10	31.57	30.48	500000
GA—FBI	Serial	Valls et al. (2005)	35.39	33.24	31.58		
GA—DP	Both	Cervantes et al. (2008)	33.71	32.57	31.65		
GA, TS—PR	Both	Kochetov and Stolyar (2003)	34.74	33.36	32.06		
GA—FBI	Serial	Goncalves et al. (2011)	—	35.94	32.76	30.08	490320

For J120, the results in Table 5.8 are more spread than for J30 and J60. As for J60, none of the existing heuristics is able to find all the best known solutions. The proposed SS is ranked 7th, 5th and 6th for 50000, 5000 and 1000 schedules, respectively. For this instance set, the width of the confidence interval that we observed in Table 5.4 for the proposed SS (0.08, 0.04 and 0.03 point for 1000, 5000 and 50000 schedules, respectively) does not change the ranking.

Some papers conducted experiments for more than 50000 schedules. As shown in Tables 5.7 and 5.8, the results are significantly improved for J60 and J120, which should incite researchers to test metaheuristics for larger schedule limits. As these algorithms do not use the same schedule limit, the comparison of the results should be taken with caution. The proposed SS performs

better than any other metaheuristics for J60, while the obtained results are close to those obtained in the literature for J120.

As presented in Tables 5.6, 5.7 and 5.8, no heuristic provides the best results for all the schedule limits and all the project sizes. For that reason, Kolisch and Hartmann (2006) introduced the concept of dominance in order to determine the best global heuristics. A heuristic X is dominated by a heuristic Y if X has for at least one combination of instance set and schedule limit a higher average deviation than Y without having for any of the other combinations a lower average deviation. By applying this rule, the proposed SS would only be dominated by Paraskevopoulos et al. (2012) for 1000, 5000 and 50000 schedules, which makes it one of the best heuristics and confirms that hybrid metaheuristics, especially those based on scatter search, are among the best current heuristics. However, Paraskevopoulos et al. (2012) do not indicate if their results were obtained with a single run or with several runs, and, in the latter case, if their results were the best or the average among several runs. As previously explained for J60, our method is hence able to provide better results than Paraskevopoulos et al. (2012) for a single run with a schedule limit of 50000, but worst results for ten replications.

5.4.6.2 Comparative analysis with no schedule limits

Some researchers have not provided results according to schedule limits, as it is sometimes impossible to count the number of generated schedules. These methods are presented in Tables 5.9, 5.10 and 5.11 for J30, J60 and J120, respectively. The first column describes each heuristic: PCAP for Parallel Complete Anytime Procedure, FF for Filter-and-Fan approach, LSSPER for Local Search with Subproblem Exact Resolution, VNS for Variable Neighborhood Search, PBP for Population Based Procedures, DBGA for Decomposition Based GA, MP for Multi-Pass approach, SA for Simulated Annealing, ATLAS for Accelerating Two-Layer Anchor Search, PASS for Polarized Adaptive Scheduling Scheme, LR for Lagrange Relaxation. Since the reported results differ in terms of stopping criterion and computational effort, it is hard to carry out any comparison with the proposed SS. The analysis presented hereinafter should then be taken with caution.

Table 5.9: Computational comparison with an unlimited number of schedules - J30

Algorithm	SGS	Authors	Avg. dev. (%)	No. of schedules		CPU(s)		CPU freq.
				Avg.	Max.	Avg.	Max.	
SS—FBI	**Serial**	**This study (50000 sched.)**	**0.00**	**—**	**—**	**3.50**	**191.02**	**2.53 GHz**
PCAP	—	Zamani (2010b)	0.00	—	—	4.09	—	1.86 GHz
FF	Serial	Ranjbar (2008)	0.00	—	—	5.00	—	3 GHz
LSSPER	Serial	Palpant, Artigues and Michelon (2004)	0.00	830	1120	10.26	123.00	2.3 GHz
VNS	Serial	Fleszar and Hindi (2004)	0.01	—	—	0.64	5.86	1 GHz
SS—FBI	**Serial**	**This study (5000 sched.)**	**0.02**	**—**	**—**	**0.80**	**15.19**	**2.53 GHz**
TS—FBI	Serial	Valls et al. (2003)	0.06	—	—	1.61	6.15	400 MHz
SS—FBI	**Serial**	**This study (1000 sched.)**	**0.10**	**—**	**—**	**0.27**	**2.80**	**2.53 GHz**
PBP	Serial	Valls et al. (2004)	0.10	—	—	1.16	5.49	400 MHz
DBGA	Serial	Debels and Vanhoucke (2007)	0.12	—	—	0.01	—	1.8 GHz
Netw. Decomp.	—	Sprecher (2002)	0.12	—	—	2.75	39.7	166 MHz
GA	Serial	Hindi, Yang and Fleszar (2002)	0.37	1683	3068	0.17	0.49	400 MHz
MP—netw. flow	Paral.	Artigues et al. (2003)	1.74	30	30	—	—	—

For J30, Table 5.9 shows that the proposed SS produces better results in less time than these algorithms, except the GA of Hindi et al. (2002), the VNS of Fleszar and Hindi (2004) and the DBGA of Debels and Vanhoucke (2007). The latter achieved a worse average deviation than the one obtained with the proposed SS for 1000 schedules, but with a significantly faster computation time. For 5000 schedules, Fleszar and Hindi (2004) obtained a better average deviation with a slightly lower computation time. As presented in Table 5.10 and Table 5.11, Thammano and Phu-ang (2012) obtained competitive results but they did not provide the computational times and the numbers of generated schedules. For J60, Ranjbar (2008) produced a better average deviation than the proposed SS for 50000 schedules, with a better computational time. Debels and Vanhoucke (2007) and Zamani (2012) obtained a worse average deviation compared to the proposed SS for 50000 schedules, but with better computational times. Valls et al. (2004) obtained a slightly better average deviation with a slightly faster computation time. Table 5.11 highlights that Debels and Vanhoucke (2007) produced better results than the proposed SS for J120 and 50000 schedules in terms of average deviation and computation time, while Valls et al. (2004), Ranjbar (2008) and Zamani (2012) produced competitive results very fast. From this analysis, it can be concluded that none of these heuristics globally perform better than the proposed SS in terms of both computation time and solution quality for all the instance sets.

Table 5.10: Computational comparison with an unlimited number of schedules - J60

Algorithm	SGS	Authors	Avg. dev. (%)	No. of schedules		CPU(s)		CPU freq.
				Avg.	Max.	Avg.	Max.	
SS—FBI	**Serial**	**This study (50000 sched.)**	**10.45**	**—**	**—**	**735.60**	**12800.24**	**2.53 GHz**
GA—TS—SA	Serial	Thammano and Phu-ang (2012)	10.52	—	—	—	—	—
FF	Serial	Ranjbar (2008)	10.56	—	—	5.00	—	3 GHz
SS—FBI	**Serial**	**This study (50000 sched.)**	**10.58**	**—**	**—**	**34.66**	**369.89**	**2.53 GHz**
PASS	Both	Zamani (2012)	10.64	—	—	3.00	—	1.86 GHz
DBGA	Serial	Debels and Vanhoucke (2007)	10.68	—	—	1.11	—	1.8 GHz
ATLAS	Both	Zamani (2010a)	10.81	—	—	36.20	—	1.86 GHz
LSSPER	Serial	Palpant et al. (2004)	10.81	1622	2262	38.78	223	2.3 GHz
PBP	Serial	Valls et al. (2004)	10.89	—	—	3.65	22.60	400 MHz
SS—FBI	**Serial**	**This study (5000 sched.)**	**10.93**	**—**	**—**	**3.24**	**26.42**	**2.53 GHz**
VNS	Serial	Fleszar and Hindi (2004)	10.94	—	—	8.89	80.70	1 GHz
PCAP	—	Zamani (2010b)	11.07	—	—	28.45	—	1.86 GHz
SS—FBI	**Serial**	**This study (1000 sched.)**	**11.38**	**—**	**—**	**0.90**	**5.32**	**2.53 GHz**
TS—FBI	Serial	Valls et al. (2003)	11.45	—	—	2.76	14.61	400 MHz
Netw. Decomp.	—	Sprecher (2002)	11.61	—	—	460.16	4311.46	166 MHz
TS—netw. flow	Paral.	Artigues et al. (2003)	12.05	—	—	3.20	—	—
MP—netw. flow	Paral.	Artigues et al. (2003)	14.20	60	60	—	—	—
LR—activity list	Both	Möhring, Schulz, Stork and Uetz (2003)	15.60	—	—	6.9	57	200 MHz

Table 5.11: Computational comparison with an unlimited number of schedules - J120

Algorithm	SGS	Authors	Avg. dev. (%)	No. of schedules		CPU(s)		CPU freq.
				Avg.	Max.	Avg.	Max.	
SS—FBI	**Serial**	**This study (50000 sched.)**	**30.39**	**—**	**—**	**2976.53**	**14179.87**	**2.53 GHz**
GA—TS—SA	Serial	Thammano and Phu-ang (2012)	30.51	—	—	—	—	—
DBGA	Serial	Debels and Vanhoucke (2007)	30.69	—	—	2.99	—	1.8 GHz
SS—FBI	**Serial**	**This study (50000 sched.)**	**31.16**	**—**	**—**	**198.14**	**647.67**	**2.53 GHz**
PASS	Both	Zamani (2012)	31.22	—	—	8.00	—	1.86 GHz
FF	Serial	Ranjbar (2008)	31.42	—	—	5.00	—	3 GHz
PBP	Serial	Valls et al. (2004)	31.58	—	—	59.43	263.97	400 MHz
LSSPER	Serial	Palpant et al. (2004)	32.41	3396	5000	207.93	501.00	2.3 GHz
ATLAS	Both	Zamani (2010a)	32.45	—	—	110.50	—	1.86 GHz
SS—FBI	**Serial**	**This study (5000 sched.)**	**32.52**	**—**	**—**	**21.04**	**83.93**	**2.53 GHz**
VNS	Serial	Fleszar and Hindi (2004)	33.10	—	—	219.86	1126.97	1 GHz
PCAP	—	Zamani (2010b)	33.78	—	—	34.23	—	1.86 GHz
SS—FBI	**Serial**	**This study (1000 sched.)**	**34.13**	**—**	**—**	**4.62**	**10.69**	**2.53 GHz**
TS—FBI	Serial	Valls et al. (2003)	34.53	—	—	17.00	43.94	400 MHz
LR—activity list	Both	Möhring et al. (2003)	36.00	—	—	72.90	654.00	200 MHz
TS—netw. flow	Paral.	Artigues et al. (2003)	36.16	—	—	67.00	—	—
Netw. Decomp.	—	Sprecher (2002)	39.29	—	—	458.53	1511.33	166 MHz
MP—netw. flow	Paral.	Artigues et al. (2003)	39.34	120	120	—	—	—

5.5 Concluding remarks

This paper presents a new hybrid metaheuristic for solving the resource-constrained scheduling project problem (RCPSP). The method is based on a scatter search framework that involves forward-backward improvement, reversing of population at each iteration of the scatter search and an advanced path relinking strategy for combining pairs of solutions. It contains two new mechanisms: a new combination method and a new reference set update method. This path relinking has two distinctive features. First, it is a bidirectional path relinking, which has been identified as one of the most promising approach for combination in scatter search. It consists in swaping the initiating and the guiding solutions at each step of the path relinking process. Second, this path relinking is based on a new move that consists in moving the most distant activity from the guiding solution in the current solution of the path to its place in the guiding solution. Finally, the reference set update method incorporate a new improvement method that enhances the quality or the diversity of the reference set.

Since the performances are highly sensitive to the values of the scatter search parameters, an advanced parameter tuning method based on local search is proposed. Computational experiments on the PSPLIB benchmark Kolisch and Sprecher (1997) show that the proposed scatter search algorithm provides high-quality solutions in reasonable computation times and that the new mechanisms provide significant improvement. An extensive comparison with the best heuristics in the literature demonstrates that the proposed approach is one of the most advanced heuristics, especially for the J30 and J60 instance sets.

We suggest three directions for further research. First, based on the obtained results, the experimental protocol commonly used in the literature should be updated by conducting several independent runs to include the effect of random devices involved in most metaheuristics, and by testing the heuristics on larger problem instances and for larger schedule limits. Second, since the hybrid metaheuristics based on scatter search are among the best heuristics, further research should be oriented to develop new mechanisms for scatter search and to include existing advanced ones for the schedule representation, the decoding procedure, the generation of the initial population, the subset generation method, the combination method, the improvement method and the use of long-term memory. Finally, the proposed hybrid scatter search could be

used to tackle other combinatorial optimization problems including variants of the standard RCPSP.

Acknowledgements

This work has been supported by the Natural Sciences and Engineering Research Council of Canada, the Jarislowsky/SNC-Lavalin Research Chair in the Management of International Projects, and the CIRRELT. This support is gratefully acknowledged.

CHAPITRE 6 ARTICLE 3: INFLUENCE OF THE PROJECT CHARACTERISTICS ON THE EFFICIENCY OF ACTIVITY OVERLAPPING

rédigé par:

François Berthaut[5], Robert Pellerin[5], Adnène Hajji[6] et Nathalie Perrier[5]

soumis pour publication dans :

Computers & Operations Research

date de soumission initiale: 7 juin 2015

Abstract

Overlapping is commonly regarded as a promising strategy to accelerate project execution. It consists in allowing downstream activities to begin before receiving all the final information required from upstream activities. However, overlapping can entail reworks in downstream activities. This raises the question of when and to which extent overlapping should be applied. This paper tackles the project scheduling problem in presence of complex networks of activities, resource constraints, overlapping and rework. The main objective is to measure and analyze the influence of eight project characteristics on the efficiency of overlapping in terms of reduction of the project makespan. The project characteristics represent the project size, the resource constraints, the overlapping opportunities and the rework. An exact solution procedure and a metaheuristic are introduced to minimize the project makespan. The exact procedure is based on a new 0-1 integer linear programming model with overlapping modes and new constraint propagation techniques as preprocessing. The metaheuristic is based on a scatter search algorithm developed for the standard Resource-Constrained Project Scheduling Problem (RCPSP).

[5] Jarislowsky/SNC-Lavalin Research Chair in the Management of International Projects and CIRRELT, Department of Mathematics and Industrial Engineering, École Polytechnique de Montreal, Montreal, Canada (francois.berthaut@polymtl.ca; robert.pellerin@polymtl.ca;nathalie.perrier@polymtl.ca)

[6] Department of Operations and Decision Systems, Faculty of Business Administration, Laval University, Québec, Canada (adnene.hajji@fsa.ulaval.ca)

Computational experiments on instances with up to 120 activities show that the exact procedure is able to solve to optimality most of the projects with 30 activities and that the metaheuristic approach produces high-quality solutions in reasonable computational time. A two-part model is used to conduct a statistical analysis of the influence of the project characteristics on the makespan gain with overlapping. The findings of the paper provide a better understanding of the overlapping decisions and should guide planners in improving existing practices.

Keywords: Activity overlapping; Concurrent engineering; Project management; Project scheduling; Makespan minimization; Linear programming; Metaheuristic; Two-part model.

6.1 Introduction

Several strategies developed to accelerate project execution, such as concurrent engineering and fast-tracking, are based on the concept of overlapping. It consists in relaxing the sequential execution of dependent activities by allowing downstream activities to begin before receiving all the final information required from upstream activities. Concurrent engineering has been demonstrated to be powerful for reducing product development times in a wide range of manufacturing industries (Terwiesch & Loch, 1999). Fast-tracking, which consists in overlapping design and build phases, is also widely applied in construction projects (Dzeng, 2006; Pena-Mora & Li, 2001). However, overlapping often causes additional reworks in downstream activities, as well as iterations of interdependent activities. Such reworks may outweigh the benefices of overlapping in terms of cost and time (Terwiesch & Loch, 1999). This raises the question of when and to which extent overlapping should be applied. In practice, project teams determine overlapping strategies on an ad hoc basis without always considering rework and interaction between activities (Lin et al., 2010).

Two groups of models have been developed in the literature to investigate this question. First, many authors have considered only one or few couples of overlappable activities and no resource constraints. Krishnan et al. (1997) proposed a model of dependency based on the concepts of upstream information evolution and downstream sensitivity. The upstream information evolution characterizes the refinement of information from its preliminary form to a final value, whereas the downstream sensitivity represents the amount of rework required to incorporate upstream changes. They also introduced a model that determine when to start the downstream activity so as

to minimize the development cycle time of a two stage process. Roemer et al. (2000) and Roemer and Ahmadi (2004) modelled the probability of rework as a function of the amount of overlap and studied the time-cost tradeoffs for several serial stages without iterations. Khoueiry et al. (2013) tackled the net benefit maximization problem of several coupled activities in parallel. Loch and Terwiesch (1998) and Lin et al. (2010) investigated the integrated problem of overlapping and information exchange policy for a two-stage process, assuming that information exchange usually requires time and cost. In addition, these authors quantified the amount of rework as a function of the amount of overlap. Finally, Tyagi et al. (2013) introduced an ant colony algorithm for solving the integrated problem of overlapping and information exchange policy for several serial stages.

Other approaches have considered overlapping in projects with complex networks of activities, using scheduling techniques. These approaches can be distinguished according to the deterministic or stochastic nature of the problem, and whether resource constraints, reworks and iterations are considered. The simulation models developed for the stochastic projects mostly use design structure matrix (DSM) to capture the information flows and the dependencies between activities. The objective is to measure the distribution of probability of the time or cost of the project. Browning and Eppinger (2002) investigated the evaluation of the project duration and gain with stochastic activity durations and used DSM to model the probabilities of rework and iteration, and the impact of reworks. The decision to start an activity is governed by a control policy depending on the system state. Also, Wang and Lin (2009) and Lim, Yi, Lee and Arditi (2014) refined the probability of rework by modelling explicitly the upstream information evolution and the downstream sensitivity, while Huang and Chen (2006) and Cho and Eppinger (2005) highlighted how resource constraints impact the overlapping decisions and delay the project completion date.

The standard deterministic Resource-Constrained Project Scheduling Problem (RCPSP) has also been extended to model overlapping without feedbacks and iterations. The objective is to find not only the final date of the project, but also the execution dates of the activities that will be used as baseline for the execution. Gerk and Qassim (2008) and Berthaut, Pellerin, Perrier and Hajji (2014) argued that even though overlapping is inherently risky, many advanced approaches developed in the literature for stochastic scheduling involve determining at a first step a baseline schedule without anticipation of uncertainty (Demeulemeester et al., 2008; Guéret & Jussien,

2008; Herroelen & Leus, 2004, 2005). Bartusch et al. (1988) and De Reyck and Herroelen (1998) extended the standard RCPSP to consider generalized precedence relations, such as finish-to-start relations with negative time lags, but the proposed models do not consider reworks. Bozejko et al. (2014) addressed the problem of road projects scheduling with resource constraints as a flow-shop scheduling problem where overlapping is allowed, but without reworks. Other academics assumed that the relation between the amount of overlap and the amount of rework is preliminary known for each couple of overlappable activities. Among them, Gerk and Qassim (2008) and Liberatore and Pollack-Johnson (2006) introduced exact methods to solve project acceleration problems based on activity crashing and overlapping with and without resource constraints, respectively.

In the aforementioned literature, only few papers have considered resource constraints and non-negligible reworks. All these papers assumed a simple linear relationship between the amount of rework and the amount of overlap which is not realistic. In order to fill this gap, Berthaut, Grèze, Pellerin, Perrier and Hajji (2011) proposed a formulation based on overlapping modes. Preliminary information can be issued at predefined intermediate points corresponding to the completion of internal milestones of the upstream activity. An expected amount of rework in the downstream activity is associated to the release of preliminary information, which is not required to be linear with the amount of overlap. If the downstream activity starts at a given milestone or before the next milestone, the same expected amount of rework is considered. The expected total amount of rework is a piecewise constant function of the amount of overlap, where each step is called an overlapping mode. Berthaut et al. (2011) proposed an exact formulation of the makespan minimization problem where overlapping is limited to the discrete sets of internal milestones of the upstream activities and information exchange is assumed to be costless and instantaneous. Grèze, Pellerin, Leclaire and Perrier (2014b) applied this method to a large set of project instances in order to analyze the effect of the number of couples of overlappable activities, the maximum amount of overlap, and the level of resource constraints. Berthaut et al. (2014) extended the problem to include coordination and communication time and costs and to consider the time-cost trade-offs. Grèze, Pellerin and Leclaire (2011) and Grèze, Pellerin, Leclaire and Perrier (2014a) released the limitation of overlapping to the discrete sets of internal milestone and proposed an exact formulation and a heuristic for the makespan minimization problem and the time-cost trade-offs. Baydoun, Haït, Pellerin, Clément and Bouvignies (2014)

developed a rough-cut capacity planning model for overlapping work packages. However, the overlapping modes are converted for convenience into activity modes in the linear programming models of the aforementioned papers, which makes the number of overlapping variable decisions very large.

In addition to the issue of modelling the reworks and the resource constraints, we identify other limitations in the current literature on deterministic project scheduling with overlapping. First, most approaches are based on exact methods that may be difficult to implement for large projects encountered in practice. The development of heuristic methods adapted to scheduling problems with overlapping is therefore crucial to provide high-quality solutions in reasonable computation time. Second, most authors present computational results on a small number of project examples. In consequence, the analysis of the overlapping decisions is specific to their examples and it is not possible to derive general insights and to capture the effects of projects characteristics.

The main objective of this paper is to measure and analyze the effects of eight project parameters on the efficiency of overlapping. The project characteristics represent the project size, the resource constraints, the overlapping opportunities and the rework. The efficiency of overlapping is measured by the relative makespan gain from the project without overlapping. The makespan and the overlapping decisions are computed by solving the RCPSP with overlapping modes with two optimization techniques. First, we introduce a new 0-1 integer linear program that directly models the overlapping modes, instead of modeling activity modes. Constraint propagation techniques are used as preprocessing to tighten the space of variables and to detect infeasibility. Second, we present an advanced scatter search-based metaheuritic derived from the high-quality metaheuristic developed by Berthaut, Pellerin, Hajji and Perrier (2015) to solve the standard RCPSP for large-sized projects. These two methods are tested on a set of instances generated with a full factorial experimental design of the project parameters. A statistical analysis of the results enables to capture the influence of the projects parameters on the efficiency of overlapping and to identify the most efficient practical overlapping strategies to improve the makespan gain. The results also provide a better understanding thumb of the best overlapping decisions and general rules of thumb are derived.

The remainder of the paper is organized as follows. Section 6.2 describes the problem statement and assumptions. The new 0-1 integer linear programming model and the constraint propagation

techniques are presented in section 6.3. The metaheuristic is developed in section 6.4. Section 6.5 presents the generator of projects with overlapping. The computational results are summarized and analyzed in section 6.6 and the findings are compared to practical overlapping strategies proposed in the literature in section 6.7. Finally, section 6.8 concludes the paper with recommendations for future work.

6.2 Problem Statement and assumptions

A project is defined by a set of activities, *S*, including two fictitious activities 0 and n+1 which correspond to the project start and end. Let d_j be the normal duration of activity j without overlapping. The symbols used throughout the paper are defined in Table 6.1. We investigate the joint optimization of the overlapping and project scheduling problems with the following assumptions:

1) The information flow is unidirectional from upstream to downstream activities.

Feedback information from downstream activities can lead to modifications in the upstream activities and cause iterations (Wang & Lin, 2009). DSM, block triangularization algorithms, aggregation and decomposition of activities can be used to determine a sequence of activities without any feedback (Browning, 2001). We assume that such preliminary studies have been conducted.

2) Preliminary information exchange is allowed between overlappable activities and is instantaneous.

The dependent activities can be categorized into non-overlappable and overlappable ones. The former are represented by classical finish-to-start precedence relations where a downstream activity requires the final output information or the completion of an upstream activity. The latter are represented by a finish-to-start-plus-lead time precedence constraint where the lead-time accounts for the amount of overlap, and thus depends on the overlapping decisions. Information exchange is assumed to require negligible time.

Table 6.1: Symbols and definitions

Symbol	Definition
n	number of non dummy activities
N_c	number of overlappable couples of activities
S	Set of activities, $S = \{0, ..., n+1\}$
d_j	Normal duration of activity j without overlapping
A	Set of overlappable couples of activities $(A_1,\ldots, A_{Nc})$
P	Set of classical finish-to-start precedence relations (no overlap is possible)
E	Set of temporal or precedence constraints $i \rightarrow k$ (i, k), $E = A \cup P$
Po_j	Set of immediate predecessors of activity j that are overlappable with activity j
Pn_j	Set of immediate predecessors of activity j that are not overlappable with activity j
P_j	Set of immediate predecessors of activity j, $P_j = Po_j \cup Pn_j$
Po	Set of activities that are overlappable with at least one direct successor
So_j	Set of immediate successors of activity j that are overlappable with activity j
Sn_j	Set of immediate successors of activity j that are not overlappable with activity j
S_j	Set of immediate successors of activity j, $S_j = So_j \cup Sn_j$
So	Set of activities that are overlappable with at least one direct predecessor
R	Set of renewable resources
R_k	Constant amount of available units of renewable resource k
R_{jk}	Per period resource requirement of activity j for renewable resource k
m_{ij}	Number of precedence/overlapping modes of couple (i, j); if not overlappable, $m_{ij} = 1$, else $m_{ij} > 1$
α_{ijm}	Amount of overlap when couple (i, j) is executed in mode m; $\alpha_{ij1} = 0$ (no overlap)
r_{ijm}	Amount of rework in the downstream activity j when couple (i, j) is executed in mode m, $r_{ij1} = 0$
T	Upper bound of the project makespan
$t = \{0, ..., T\}$	Periods
ES_j, LS_j	Earliest and latest possible start time of activity j, respectively
EF_j, LF_j	Earliest and latest possible finish time of activity j, respectively
$milestone_j^a$	time of the milestone of activity j for a progress of a%

3) Overlapping can be executed according to overlapping modes.

The activity progress is measured in practice according to the completion of internal milestones which are major events, such as design criteria frozen, detailed design completed, drawings finalized, or any activity deliverable. This preliminary information is used as input for downstream activities. The rework caused by overlapping on the downstream activities is defined for each milestone of the upstream activity and remains constant between two consecutive

milestones. These overlapping modes defined in Berthaut et al. (2011) can be seen as different overlapping configurations from no overlapping to aggressive overlapping.

4) The problem is formulated in a deterministic environment.

Scheduling is performed on a period-by-period basis: resource availabilities and allocations are estimated per period, while activity durations and the amounts of rework and overlap are discrete multiples of one period (Hartmann, 1999). We assume that the expected amount of rework is preliminary known for each overlapping mode of each couple of overlappable activities. The problem does not address schedule risks. As some of the most advanced approaches developed in the literature for stochastic scheduling involve determining first a baseline schedule without anticipation of uncertainty (Demeulemeester et al., 2008; Guéret & Jussien, 2008; Herroelen & Leus, 2004, 2005), the formulation in a deterministic environment constitutes a first step towards the development of approaches for stochastic scheduling with overlapping.

6.2.1 Overlapping process model with precedence and overlapping modes

The concept of overlapping modes introduced by Berthaut et al. (2011) is used in the present paper. If the downstream activity starts at a given milestone of the upstream activity or before the next milestone, the same expected amount of rework is considered. The expected amount of rework is then a piecewise constant function of the amount of overlap where each step is called an overlapping mode. Several authors have explored the behaviour and interactions of two overlapped activities and have quantified the amount of rework as a function of the amount of overlap (Lin et al., 2010; Loch & Terwiesch, 1998; Roemer et al., 2000). They highlighted that the total amount of rework is a convex increasing function of the amount of overlap. However, the models proposed in the literature for projects with a complex network and resource constraints consider a simplistic linear relation between the amount of rework and the amount of overlap (Cho & Eppinger, 2005; Gerk & Qassim, 2008). The overlapping model proposed here relaxes this assumption and can encompass any overlapping models proposed so far for two activities.

The overlapping process of two dependent activities (i, j) in A with overlapping modes is depicted in Figure 6.1. The downstream activity j starts with preliminary inputs from activity i.

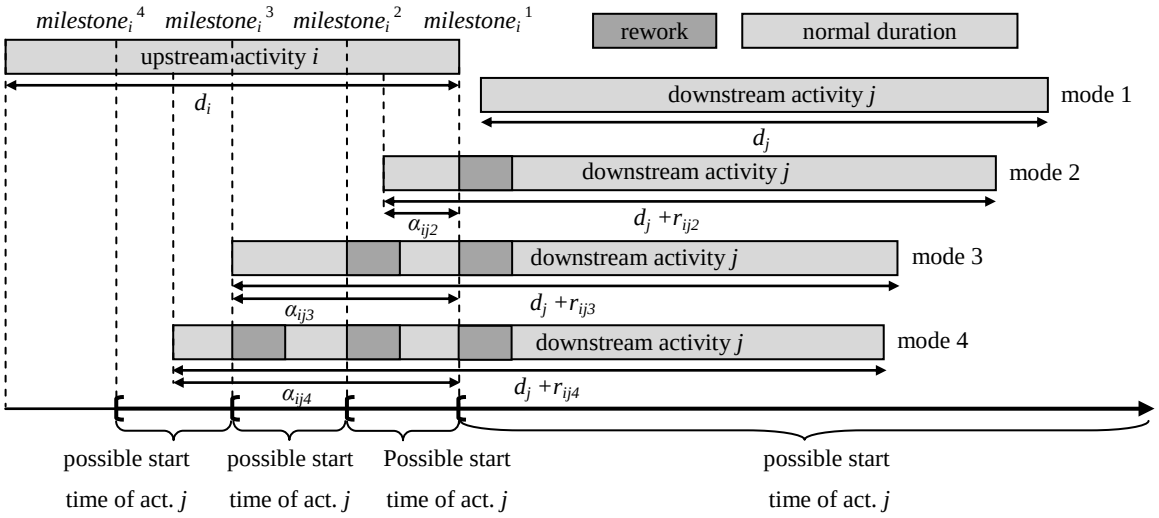

Figure 6.1: Overlapping process with overlapping modes

The amount of overlap α_{ijm} in mode $m \in \{2, ..., m_{ij}\}$ is defined as the difference between the finish time of activity i and the start time of activity j. As the upstream activity proceeds, its information evolves to its final form and is released at intermediate points to the downstream activity j. The expected amount of rework in the downstream activity is denoted by r_{ijm} in mode $m \in \{1, ..., m_{ij}\}$ with $r_{ijm} \geq 0$. The concept of overlapping modes can be generalized to precedence/overlapping modes in order to describe all precedence constraints $(i, j) \in E$. For each precedence constraint $(i, j) \in P$, there is only one mode ($m_{ij} = 1$) with $\alpha_{ij1} = 0$ and $r_{ij1} = 0$. If $(i, j) \in A$, there exist additional overlapping modes associated with the milestones of the upstream activity ($m_{ij} > 1$). The precedence constraint of two overlappable activities (i, j) depends on the overlapping mode: when not overlapped ($m = 1$), the downstream activity start time is superior or equal to the upstream activity finish time, whereas the downstream activity start time belongs to the interval $[milestone_i^{\,m}, milestone_i^{\,m-1}[$ in the case of overlapping in mode $m \in \{2, ..., m_{ij}\}$.

There is no restriction concerning the number of overlappable predecessors. If a downstream activity can be overlapped by several upstream activities, the amount of rework in downstream activity is assumed to be the sum of the amounts of rework caused by each upstream activity, as assumed in Cho and Eppinger (2005). If an activity j is both the upstream activity for a couple (j, l) and the downstream activity for another couple (i, j), we assumed that activity l must start after the end of activity i in order to eliminate the influence of information change in activity i on

activity *l*. This assumption is referred to as "sashimi-style" overlapping in Imai et al. (1985) and Roemer and Ahmadi (2004).

6.3 Exact procedure for the RCPSP with overlapping modes

In this section, we present a new 0-1 integer linear programming model of the project makespan minimization problem with overlapping modes. In Berthaut et al. (2011), Berthaut et al. (2014), Grèze et al. (2011), Grèze et al. (2014a), Grèze et al. (2014b) and Baydoun et al. (2014), the overlapping modes were converted for convenience into activity modes, which represent all the combinations of overlapping modes of an activity with the associated overlappable activities. However, the total number of activity modes and decision variables can grow substantially for an activity with several overlappable predecessors or successors. The model proposed below makes directly use of the overlapping modes. In addition, propagation constraint techniques inspired from the standard RCPSP are proposed. These techniques can be used as preprocessing to tighten the space of variables and to detect infeasibility of the problem for a given upper bound of the project makespan.

6.3.1 0-1 integer linear programming model

Each activity j in S must finish within the time window $\{EF_j, ..., LF_j\}$. The proposed model is derived from the well-known model introduced by Pritsker et al. (1969). We define the decision variables as follows:

$$X_{jt} = \begin{cases} 1 & \text{if activity } j \text{ finishes at time } t \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \in S,\ \forall t \in \{0..T\} \tag{1}$$

$$c_{ijm} = \begin{cases} 1 & \text{if } (i,j) \text{ is executed in mode } m \\ 0 & \text{otherwise} \end{cases} \qquad \forall (i,j) \in A,\ \forall m \in \{1..m_{ij}\} \tag{2}$$

$$U_{jt} = \begin{cases} 1 & \text{if time } t \text{ is superior to the the start date of activity } j \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \in So,\ \forall t \in \{0..T\} \tag{3}$$

The X_{jt} and c_{ijm} determine the execution dates and the overlapping decisions, respectively. U_{jt} is a binary variable introduced to linearize the optimization model. The main objective is to find a

precedence, overlapping and resource-feasible schedule that minimizes the project makespan. A second objective is added to limit overlapping in case of tie. The problem is formulated as follows:

$$\text{Minimize} \sum_{t=EF_{n+1}}^{t=LF_{n+1}} t \cdot X_{n+1t} + \frac{1}{2}\left(\frac{\sum_{(i,j)\in A} \sum_{m=1}^{m=m_{ij}} c_{ijm} \cdot r_{ijm}}{\sum_{(i,j)\in A} r_{ijm_{ij}}} + \frac{\sum_{(i,j)\in A} \sum_{m=1}^{m=m_{ij}} m \cdot c_{ijm}}{\sum_{(i,j)\in A} m_{ij}} \right) \quad (4)$$

Subject to:

$$\sum_{t=EF_i}^{t=LF_i} t \cdot X_{it} \le \sum_{t=EF_j}^{t=LF_j} t \cdot X_{jt} - d_j - \sum_{b\in Po_j} \sum_{m=2}^{m=m_{bj}} c_{bjm} \cdot r_{bjm}\,,\ \forall (i,j) \in P \quad (5)$$

$$\sum_{t=EF_i}^{t=LF_i} t \cdot X_{it} - \sum_{m=2}^{m=m_{ij}} c_{ijm} \cdot \alpha_{ijm} \le \sum_{t=EF_j}^{t=LF_j} t \cdot X_{jt} - d_j - \sum_{b\in Po_j} \sum_{m=2}^{m=m_{bj}} c_{bjm} \cdot r_{bjm}\,,\ \forall (i,j) \in A \quad (6)$$

$$\sum_{t=EF_i}^{t=LF_i} t \cdot X_{it} - 1 - \sum_{m=2}^{m=m_{ij}} c_{ijm} \cdot \alpha_{ijm-1} \ge \sum_{t=EF_j}^{t=LF_j} t \cdot X_{jt} - d_j - \sum_{b\in Po_j} \sum_{m=2}^{m=m_{bj}} c_{bjm} \cdot r_{bjm} - \left(LS_j - EF_i + 1\right) \cdot c_{ij1} \quad (7)$$
$$\forall (i,j) \in A$$

$$\sum_{t=EF_i}^{t=LF_i} t \cdot X_{it} \le \sum_{t=EF_l}^{t=LF_l} t \cdot X_{lt} - d_l - \sum_{b\in Po_l} \sum_{m=2}^{m=m_{bl}} c_{blm} \cdot r_{blm}\,,\ \forall j \in Po \cap So\,,\ \forall i \in Po_j\,,\ \forall l \in So_j \quad (8)$$

$$t - \sum_{b=EF_j}^{b=LF_j} b \cdot X_{jb} + d_j + \sum_{i\in Po_j} \sum_{m=1}^{m=m_{ij}} c_{ijm} \cdot r_{ijm} \le \left(T - ES_j + 1\right) \cdot U_{jt}\,,\ \forall j \in So\ ,\ \forall t \in \{0..T\} \quad (9)$$

$$\sum_{j\in S\setminus So} R_{jk} \cdot \sum_{b=\max(t,EF_j)}^{b=\min(t+d_j-1,LF_j)} X_{jb} + \sum_{i\in So} R_{jk} \cdot \left(U_{it} - \sum_{b=EF_i+1}^{b=t} X_{i,b-1} \right) \le R_k\,, \qquad \forall k \in R\,,\ \forall t \in \{0..T\} \quad 10)$$

$$\sum_{t=EF_j}^{t=LF_j} X_{jt} = 1,\ \forall j \in S \quad 11)$$

$$\sum_{m=1}^{m=m_{ij}} c_{ijm} = 1, \ \forall (i,j) \in A \qquad 12)$$

$$X_{jt} = \{0,1\}, \qquad \forall j \in S, \forall t \in \{0..T\} \qquad 13)$$

$$c_{ij} = \{0,1\}, \ \forall (i,j) \in A \qquad 14)$$

$$U_{jt} = \{0,1\}, \ \forall j \in So, \ \forall t \in \{0..T\} \qquad 15)$$

The first part of the objective function (4) represents the project makespan. The second part is a global measure of the degree of overlapping, composed of the ratio between the sum of the amounts of rework and the sum of the maximum possible amounts of rework, and the ratio between the sum of the modes and the sum of the maximum modes. Constraints (5) represent the finish-to-start precedence constraints for the activities that cannot overlap. Constraints (6) and (7) reflect the constraints depicted in Figure 6.1. If (*i, j*) is overlapped in mode $m > 2$ then constraints (6) and (7) guarantee that the start time of activity *j* belongs to the interval [$milestone_i^m$, $milestone_i^{m-1}$[. If (*i, j*) is not overlapped ($c_{ij1} = 1$), then constraints (7) are not restrictive and constraints (6) represent classical finish-to-start precedence constraints. Constraints (8) represent the "sashimi-style" overlapping assumption. Constraints (9) state that the additional variable U_{jt} is equal to 1 when *t* is superior to the start date of activity *j* in *So*. Consequently, the expression with parentheses in Constraints (10) is equal to 1 for all the periods where activity *j* is executed. Constraints (10) guarantee that the resource consumption does not exceed the resource capacity. Constraints (11) and (12) ensure that each activity and each couple of overlappable activities is assigned only one finish time and one overlapping mode, respectively. Finally, constraints (13) to (15) define the binary decision variables.

6.3.2 Constraint propagation as preprocessing

The most basic way to derive the time windows $\{EF_j, ..., LF_j\}$ for a given upper bound is to apply a modified forward and backward recursion algorithms that includes overlapping. However, this method overlooks the resource constraints. To illustrate how resource constraints can be involved, consider two independent activities $(i, j) \notin E$ such that the sum of the required

resources exceeds the resource capacities. Obviously, these activities cannot be executed in parallel. In addition, if their time windows are such that $LF_i - ES_j < d_i + d_j$, then we can deduce that activity i must be executed before activity j. By applying again the forward and backward recursion algorithms, the time windows may be tightened. In order to include the resource constraints, such constraint propagation can be used as preprocessing technique to deduce new precedence constraints and tighten the time windows. In addition, infeasibility for a given upper bound can be detected. Such techniques have been successfully applied on the standard RCPSP to compute good quality lower bounds and to prune the search tree in branch and bound exact procedures (Brucker & Knust, 2012). Constraint propagation has also been extended to the multi-mode RCPSP to eliminate superfluous activity modes (Brucker & Knust, 2012; Sprecher et al., 1997). In this section, we present a constraint propagation algorithm adapted to the RCPSP with overlapping. The algorithm maintains and evolves the sets (C, N, D, B, L, V, W) for a given upper bound T until no improvement can be done.

1) Definition of the sets (C, N, D, B, L, V, W) used by the Constraint propagation techniques

Given a schedule represented by the vectors of the start times (S_0, ..., S_{n+1}) and finish times (F_0, ..., F_{n+1}), each pair of activities $(i, j) \in S^2$ belongs exactly to one of the three following sets:

- the set C of conjunctions ($i \rightarrow j$) : if activity j does not start before i is finished ($F_i \leq S_j$),
- the set N of parallelity relations ($i \parallel j$): if activities i and j are processed in parallel ($F_i > S_j$ and $F_j > S_i$),
- the set D of disjunctions (i - j): if activities i and j cannot be executed in parallel: $F_i \leq S_j$ or $F_j \leq S_i$.

The sets (L, V, W) contain the information about the feasible overlapping opportunities:

- the overlapping matrix L indicates for each couple $(i, j) \in A$ the minimum and maximum feasible modes (m_{ij}^{min}, m_{ij}^{max}, resp.), the amounts of rework (r_{ij}^{min}, r_{ij}^{max}, resp.) and the amounts of overlap (α_{ij}^{min}, α_{ij}^{max}, resp.),
- the matrix V of minimum and maximum rework contains for each activity $i \in So$ the minimum and maximum total amount of rework (r_i^{min}, r_i^{max}, resp.) caused by all overlappable predecessors,

- the set W of pairs of couples of overlappable activities that cannot overlap at the same time contains the pairs with the same upstream activity $((i_1, j), (i_2, j))$ or the same downstream activity $((i, j_1), (i, j_2))$.

Following the formalism of Brucker and Knust (2000) and Demassey et al. (2005), we define a start-to-start matrix $B = (b_{ij})_{S \times S}$. The main interest of this matrix is to reflect the information of the other sets and to evolve with them by propagating the constraints. A start to start matrix is any integer matrix that satisfies the following conditions:

$$S_j - S_i \geq b_{ij}, \forall (i, j) \in S^2 \tag{16}$$

2) Initialization of the sets (C, N, D, B, L, V, W) with ($C_0, N_0, D_0, B_0, L_0, V_0, W_0$)

The time-window vectors LS, LF, ES, EF are first computed by applying a modified forward and backward recursion algorithms that includes overlapping for an upper bound T. (C, N, D, L, V, W) can then be initialized with (C_0, N_0, D_0, L_0, V_0, W_0) such that:

- C_0 is given by the set of all precedence relations $(i, j) \in P$, and the set $\left\{ (i,l) \in S^2 \mid \exists\, j \in Po \cap So \text{ with } i \in Po_j \text{ and } l \in So_j \right\}$ (see constraints (8),
- D_0 is given by the set of all pairs (i, j) that cannot be executed in parallel due to required resources exceeding the resource capacity (i,e., $r_{ik} + r_{jk} > R_k$ for any $k \in R$),
- N_0 is the set of all pairs (i, j) such that
  $d_i + d_j + r_i^{\min} + r_j^{\min} > \max\{LF_i, LF_j\} - \min\{ES_i, ES_j\}$,
- L_0, V_0, and W_0 are defined with all the modes and reworks of all couples $(i, j) \in A$.

Inspired from the start-to-start matrix developed by Brucker and Knust (2003) and Brucker and Knust (2012) for the multi-mode RCPSP with time lags, B_0 can be initialized as follows:

$$b_{ij} = \begin{cases} 0 & \text{if } i = j \\ d_i & \text{if } i \rightarrow j \\ -(d_j - 1) & \text{if } i \parallel j \\ d_i - \alpha_{ij}^{\max} & \text{if } (i,j) \in A \\ -(T - d_i) & \text{otherwise} \end{cases} \quad , \forall (i,j) \in S^2 \tag{17}$$

The propagation of these initial inequalities can be performed with the Floyd-Marshall algorithm (Brucker, Knust, Schoo, & Thiele, 1998; Demassey et al., 2005) which computes the transitive closure of B_0 in $O(n^3)$. b_{0i} and $-b_{i0}$ are the early start and late start of activity i, respectively. We can subsequently strengthen B by setting $b_{i0} = -LS_i$ and $b_{0i} = ES_i$ to take into account more carefully the overlapping modes.

3) Constraint propagation algorithm

The proposed algorithm is an iterative process involving the following methods to deduce new relations:

- Path consistency: Every time a new relation is deduced, B is adjusted by applying the Floyd-Marshall algorithm and by computing the new time windows with the forward and backward recursion algorithms, considering the feasible overlapping opportunities and the set of conjunctions.
- The immediate selection algorithm (Carlier & Pinson, 1989): It replaces each disjunction (i - j) by the precedence constraint ($i \rightarrow j$) whenever $b_{ij} \geq 1 - d_j - r_j^{min}$. If $(i, j) \in A$, then the matrix L can be updated to keep $m = 1$ as the only feasible mode.
- The Symmetric triples rules (Brucker et al., 1998): A triple of activities (i, j, k) is called symmetric if $k \parallel i$ and $k \parallel j$ hold and i, j, k cannot be processed simultaneously because of resource constraints. If (i, j, k) is a symmetric triple, then i and j are in disjunction (i.e., i - j). Other relations can be deduced by considering an additional activity l in relation with a symmetric triple (i, j, k).
- Edge-Finding rules (Demassey et al., 2005): These rules make use of the concept of clique of disjunctions, which is a set of activities such that no pair of activities (i, j) in the set can be executed in parallel (i.e., (i, j) is either a conjunction or a disjunction). The

edge-finding rules detect whether an activity of the set has to be executed after or before all the activity of the cliques. We denote by $G = \{G_1, \ldots, G_q\}$ the set of cliques, which are computed with the heuristics proposed in Brucker et al. (1998) and Baptiste and Le Pape (2000).

– Shaving techniques: These tests determine whether there exists a solution that satisfies a given condition. A new constraint is temporarily added and constraint propagation is performed. If it leads to infeasibility, then the opposite constraint is valid. Since shaving is very time-consuming (Demassey et al., 2005), this technique is applied in our procedure only on the sets D and W. For each disjunction (i - j) in D, we test if adding $i \rightarrow j$ or $j \rightarrow i$ leads to infeasibility. If no infeasibility is detected, the start-to-start matrix B can be updated by $B = \min(B^{i \rightarrow j}, B^{i \rightarrow j})$, where $B^{i \rightarrow j}$ and $B^{i \rightarrow j}$ represent the start-to-start matrix obtained after adding $i \rightarrow j$ and $j \rightarrow i$, respectively. Similarly, for each pair ((i, j), (k, l)) in W, we test if setting $m_{ij} = 1$ or $m_{kl} = 1$ leads to infeasibility. If no infeasibility is detected, the start-to-start matrix B can be updated by $B = \min(B_{ij}, B_{kl})$, where B_{ij} and B_{kl} represent the start-to-start matrix obtained after setting $m_{ij} = 1$ and $m_{kl} = 1$, respectively.

The aforementioned rules already exist in the literature, although they have been modified to take into account overlapping. In addition, the following rules specific to overlapping are introduced:

– If $(i, j) \in A$ is shown to be a conjunction (i.e., $(i, j) \in A \cap C$), then the only feasible mode is $m = 1$ and we can set $m_{ij}^{min} = m_{ij}^{max} = 1$, $r_{ij}^{min} = r_{ij}^{max} = 0$ and update r_j^{min} and r_j^{max}.

– If $(i, j) \in A$ is shown to be a parallelity relation (i.e., $(i, j) \in N \cap A$), then the minimal amount of overlap α_{ij}^{min} in matrix L, and consequently the minimal mode m_{ij}^{min} and the minimal amounts of rework r_{ij}^{min} and r_j^{min} can be limited to those that respect the following conditions:

$$d_i + r_i^{min} + d_j - \alpha_{ij} + r_{ijm} \leq LF_j - ES_i \quad \forall\, m \in [m_{ij}^{min}, m_{ij}^{max}] \text{ where } \alpha_{ij} \in [\alpha_{ijm}, \alpha_{ijm}\text{-}1[\qquad (18)$$

– If $(i, j) \in A$, then the minimal and maximal amounts of overlap α_{ij}^{min} and α_{ij}^{max} in matrix L, and consequently m_{ij}^{min}, m_{ij}^{max}, r_{ij}^{min}, r_{ij}^{max}, r_j^{min} and r_j^{max}, must respect the two following conditions:

$$\alpha_{ij}^{min} \geq d_i + r_i^{min} + b_{ji}$$
$$\alpha_{ij}^{max} \leq d_i + r_i^{max} - b_{ij} \quad (19)$$

– If $i \in So$, then the maximum total amount of rework r_i^{max} in matrix V, and consequently the maximal amount of rework r_i^{max} for each overlappable upstream activity $l \in Po_i$, must respect the following condition:

$$r_i^{max} \leq LF_i - ES_i - d_i \quad (20)$$

– Let $i \in So$ be a downstream activity for more than one couple of overlappable activities. If any pair (j, k) of Po_i^2 is a disjunction (j - k) or a conjunction ($j \rightarrow k$ or $k \rightarrow j$), then only one of the couples of overlappable activities (j, i) and (k, i) can overlap. These two couples are added to the set W.

– Let $i \in Po$ be an upstream activity for more than one couple of overlappable activities. If any pair (j, k) of So_i^2 is a disjunction (j - k) or a conjunction ($j \rightarrow k$ or $k \rightarrow j$), then only one of the couples of overlappable activities (i, j) and (i, k) can overlap. These two couples are added to the set W.

The constraint propagation algorithm stops when no more adjustments can be performed or when infeasibility is detected (i.e., whenever $b_{ii} > 0$ holds for some activity i). The latter means that $T + 1$ constitutes a lower bound of the problem.

6.3.3 Valid inequalities inferred from constraint programming

If infeasibility has not been detected, the 0-1integer linear programming model presented in section 6.3.1 can be enhanced to include the information derived from the constraint propagation algorithm. The modified model uses the tightened time-windows $\{EF_j, ..., LF_j\}$ and include the following constraints:

$$\sum_{m=m_{ij}^{min}}^{m=m_{ij}^{max}} c_{ijm} = 1, \forall (i, j) \in A \quad (21)$$

$$\sum_{t=EF_i}^{t=LF_i} t \cdot X_{it} - \left(\sum_{t=EF_j}^{t=LF_j} t \cdot X_{jt} - d_j - \sum_{b \in Po_j} \sum_{m=2}^{m=m_{bj}} c_{bjm} \cdot r_{bjm} \right) \geq \alpha_{ij}^{\min}, \forall (i,j) \in A \quad (22)$$

$$\sum_{t=EF_i}^{t=LF_i} t \cdot X_{it} - \left(\sum_{t=EF_j}^{t=LF_j} t \cdot X_{jt} - d_j - \sum_{b \in Po_j} \sum_{m=2}^{m=m_{bj}} c_{bjm} \cdot r_{bjm} \right) \leq \alpha_{ij}^{\max}, \forall (i,j) \in A \quad (23)$$

$$\sum_{j \in G_h \cap S \setminus So} \sum_{b=\max(t, EF_j)}^{b=\min(t+d_j-1, LF_j)} X_{jb} + \sum_{i \in G_h \cap So} \left(U_{it} - \sum_{b=EF_i+1}^{b=t} X_{i,b-1} \right) \leq 1, \forall G_h \in G, \ \forall t \in \{0..T\} \quad (24)$$

$$\sum_{j \in D_h \cap S \setminus So} \sum_{b=\max(t, EF_j)}^{b=\min(t+d_j-1, LF_j)} X_{jb} + \sum_{i \in D_h \cap So} \left(U_{it} - \sum_{b=EF_i+1}^{b=t} X_{i,b-1} \right) \leq 1, \forall D_h \in D, \ \forall t \in \{0..T\} \quad (25)$$

$$\sum_{m=2}^{m=m_{ij}} c_{ijm} + \sum_{w=2}^{w=m_{ikl}} c_{iklw} = 1, \forall ((i,j),(k,l)) \in W \quad (26)$$

Constraints (12) are replaced by constraints (21) to limit the problem to the identified feasible modes. Constraints (22) and (23) guarantee that the amount of overlap is between α_{ij}^{min} and α_{ij}^{max}. Constraints (24) state at most one activity is processed at any time for each clique. Similarly, Constraints (25) ensure that at most one activity is processed at any time for each pair of activities in *D*. Constraints (26) state that at most one couple of overlappable activities can overlap in the set *W*. In addition, the precedence constraints (5) can be defined for all the conjunctions in *C* instead of all the initial precedence relations in *P*. The set *A* can also be adjusted to eliminate the couples that cannot overlap according to the constraint propagation algorithm.

To sum up, the constraint propagation algorithm is used as preprocessing to tighten the time-windows of the activities and to deduce additional inequalities for the proposed 0-1 integer linear programming model for a given horizon *T*. If infeasibility is detected, then *T* + 1 constitutes a lower bound of the problem. Otherwise, the 0-1 integer linear programming model composed of the objective (4) and the constraints (5-11, 13-15, and 21-26) is used to solve the RCPSP with overlapping.

6.4 A path relinking-based scatter search for the RCPSP with overlapping

The problem under study is an extension of the standard RCPSP, which belongs to the class of NP-hard optimization problems (Blazewicz et al., 1983). Most projects in practice have a size larger than one hundred activities (Icmeli-Tukel & Rom, 1997; Liberatore et al., 2001). However, exact procedures for the RCPSP have only the capability to solve small scale projects with at most sixty activities in acceptable computation effort (Herroelen, 2005). This limitation has motivated a growing number of academics to develop heuristics that are able to find good-quality schedules in reasonable computation time (Hartmann & Kolisch, 2000; Kolisch & Hartmann, 2006). Following this trend, Berthaut et al. (2015) developed a path relinking-based scatter search algorithm for the standard RCPSP with two new mechanisms. First, a bidirectional path relinking method with a new move was used as combination method. Second, a new improvement procedure is proposed in the reference set update method. The computational results showed that the proposed method is among the best performing metaheuristics. We propose in this section an extension of this metaheuristic that incorporates the overlapping modes.

6.4.1 General structure of the Path Relinking-based Scatter search

Introduced by Glover (1977), scatter search (SS) is an evolutionary algorithm that generates new solutions by combining preserved ones. SS has been successfully applied to the RCPSP by Valls et al. (2004), Debels et al. (2006), Ranjbar et al. (2009), Mobini et al. (2009), Chen et al. (2010), Paraskevopoulos et al. (2012) and Berthaut et al. (2015). The algorithm proposed in this paper for the RCPSP with overlapping is depicted in Figure 6.2. An initial population of size *InitPop* is first generated. A reference set *RefSet* composed of two distinct sets of high-quality and diversified solutions is build from the population of solutions. The reference solutions will be evolved to form a new population *New_Pop*. The new population is initialized with the best current solution. The reference solutions are afterwards paired to form subsets that are combined with a path relinking method (PR) to generate new solutions. These solutions are evaluated and either directly added to the new population or first improved by forward-backward improvement (FBI) depending on their quality. The algorithm stops when the number of generated schedules

reaches *NSched_limit*. At each iteration, the scheduling direction and the project network are reversed. The details of each step are described below.

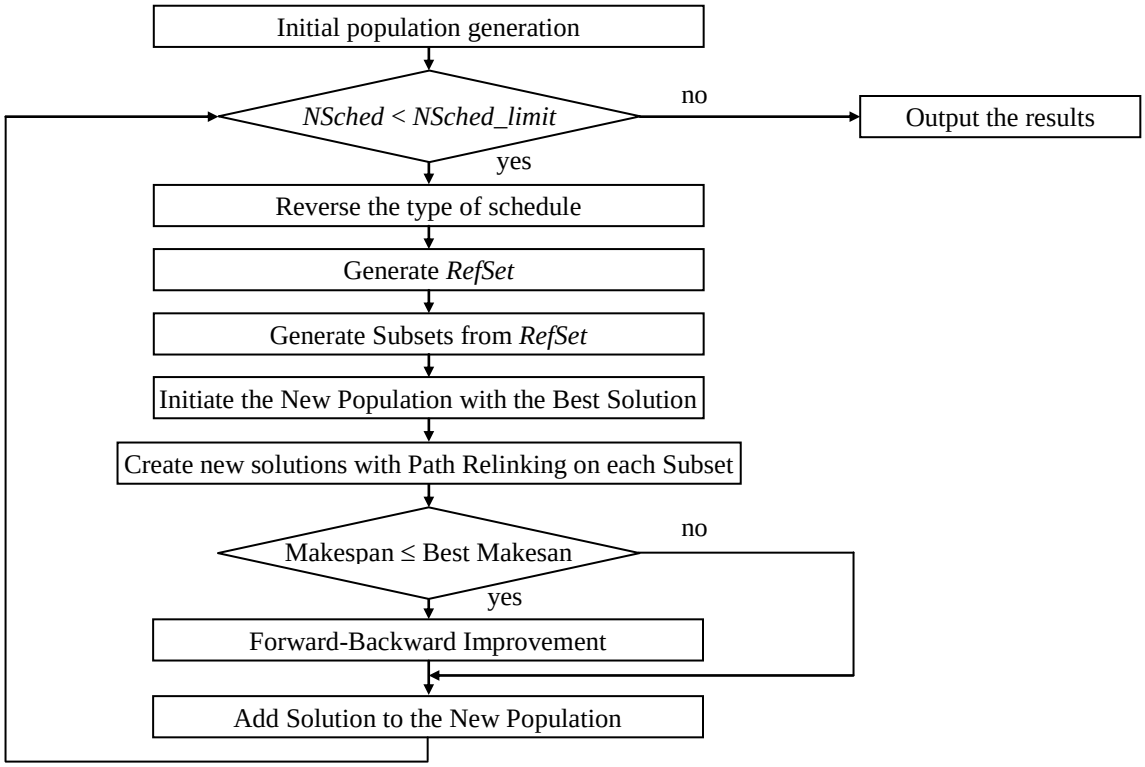

Figure 6.2: Flow chart of the proposed scatter search algorithm

6.4.2 Original and Reversed network

Ranjbar et al. (2009) introduced a SS that alternates at each iteration between original and reversed project networks in order to explore a different solution space. The reversed project network is obtained by reversing the directions of all the arrows such that the end (start) activity becomes the start (end) activity and each precedence relation $i \rightarrow j$ becomes $j \rightarrow i$. Any feasible schedule for the original (reversed) network is called a forward (backward) schedule. As both Ranjbar et al. (2009) and Berthaut et al. (2015) obtained promising results, the same mechanism is applied in the proposed SS.

6.4.3 Solution representation and decoding adapted to overlapping

The proposed algorithm is based on the well known activity list representation and the serial SGS (Kolisch & Hartmann, 1999). An activity list *AL* is a sequence of all the activities, in which the position of each activity corresponds to the order in which the activities are scheduled using a SGS. The proposed representation of schedule with overlapping decisions is composed of an *AL* and an overlapping list *OL* and is called *ALO*. We also define the position vector *P* to represent the position of each activity in *AL*. If $i = AL_h$, then the activity i is in position $p_i = h$. The overlapping list *OL* is a vector of size N_c, which assigns an overlapping mode for executing each couple of overlappable activities. *ALO* and *P* are defined as follows:

$$\begin{aligned} &ALO = (AL, OL) = ((AL_0, \ldots, AL_{n+1}), (OL_1, \ldots, OL_{Nc})) \\ &P = (p_0, \ldots, p_{n+1}) \end{aligned} \tag{27}$$

Because a single schedule can be represented by more than one activity list, Valls et al. (2003) introduced the topological order (TO) representation. We adopt the modified version of the TO representation used in Ranjbar et al. (2009) and Berthaut et al. (2015) in order to embed the TO representation into the reversing of the project network. The TO representation is adapted to generate backward (forward) children from forward (backward) solutions. A forward (backward) *ALO* is first decoded with the serial SGS to obtain the finish times of the activities based on the original (reversed) project network. The activity list of *ALO* is then reordered based on the non-increasing order of their finish times: $F_i > F_j$ implies $p_i < p_j$, while $F_i = F_j$ and $i > j$ implies $p_i < p_j$. The obtained *AL* is precedence-feasible for the reversed (original) network.

Two different modified versions of the serial SGS must be developed for the forward and backward scheduling, respectively. Basically, the serial SGS starts from scratch and builds a feasible schedule by stepwise extension of a partial schedule. At each stage *j*, the activity AL_j of *AL* is selected and scheduled at the earliest precedence- and resource-feasible start time. In the serial SGS for forward scheduling, the activity AL_j of *AL* is selected at stage *j* and scheduled as in the original serial SGS if $AL_j \in S \setminus So$. If $AL_j \in So$, then we must take into account the overlapping modes and the amount of reworks. AL_j is thus scheduled at the earliest start time that respects the following conditions:

- The "sashimi-style" assumption:
  $S_{ALj} \geq F_k$, $\forall\ k \in \{k \in S \mid \exists\ i \in Po \cap So$ with $k \in Po_i$ and $AL_j \in So_i\}$,
- The precedence constraints: $S_{ALj} \geq F_k$, $\forall\ k \in Pn_{ALj}$,
- The overlapping modes defined in *OL* for the couples of overlappable activities where AL_j is a downstream activity. For each couple in $A = \{A_1, \ldots, A_{Nc}\}$ such that $A_q = (i, AL_j)$, SALj must belong to the interval $[F_i - \alpha_{ij,OLq}, F_i - \alpha_{ij,OLq\text{-}1}[$,
- The resource constraints during the execution of AL_j, including the reworks.

If it is not possible to start AL_j in $[F_i - \alpha_{ij,OLq}, F_i - \alpha_{ij,OLq\text{-}1}[$ for any couple of overlappable activities $A_q = (i, AL_j)$ with $OL_q > 1$, then we set $OL_q = OL_q - 1$ until we find a feasible starting date or until $OL_q = 1$.

In the serial SGS for backward scheduling, the activity AL_j selected at stage j is scheduled as in the original SGS with the reversed precedence relations (i.e., $AL_j \rightarrow k$ becomes $k \rightarrow AL_j$) if $AL_j \in S \setminus So$. If $AL_j \in So$, then AL_j must be put in the partial schedule before the upstream activities by assuming that the proposed modes in *OL* are feasible. Later in the process, when the upstream activities are selected to be scheduled, it may be impossible to schedule them in the range of possible start times imposed by the proposed modes in *OL*. As for forward scheduling, the overlapping mode set in *OL* must be decreased, except that the process must go back to stage j when the downstream activity AL_j was scheduled.

6.4.4 Initial Population generation

An initial population of size *InitPop* is generated at the beginning of the SS. An *AL* and an *OL* must be generated for each member of the population. Each *OL* is generated by randomly selecting an overlapping mode between 1 and m_{ij} for each couple $(i, j) \in A$. The generation of the activity lists is governed by the regret-biased random sampling (RBRS) method with the minimum latest finish time (LFT). Each *AL* is iteratively created by randomly selecting an activity of the decision set D_k according to the following probability to be placed at position k:

$$\psi(i) = \frac{(\rho_i + 1)}{\sum_{j \in D_k} (\rho_j + 1)} \quad \text{, with } \rho_i = \max_{j \in D_k} (LF_j) - LF_i, \ \forall i \in D_k \tag{28}$$

where LF_i denotes the latest finish time of activity i calculated by the Critical Path Method (CPM) including overlapping, and the decision set D_k represents the eligible set at stage k.

6.4.5 Reverse the type of schedule

Once a new population *New_Pop* has been generated, the direction of the project network is reversed from original (reversed) to reversed (original) at the beginning of a new SS iteration. As each forward (backward) *AL* of the population has been decoded with the serial SGS, the finish times are used to obtain the topologicol order activity list for backward (forward) scheduling with the reversed (original) network. The overlapping list *OL* remains the same when the type of schedule is reversed.

6.4.6 RefSet Update Mechanism

The SS algorithm builds and maintains a subset *RefSet* of high-quality and diverse solutions found during the search. It is applied on the new population *New_Pop* right after the initial population generation or during the process once all the subsets have been combined. The reference set *RefSet* is divided into two disjoint subsets $RefSet_1$ and $RefSet_2$ of size b_1 and b_2, respectively. Each solution of $RefSet_1$ should have a distance of at least t_1 with other solutions of $RefSet_1$, while each solution of $RefSet_2$ should have a distance of at least t_2 (with $t_2 > t_1$) with any solution of $RefSet_1$ or $RefSet_2$. The distance measure between two solutions represented by $ALO_1 = (AL_1, OL_1)$ and $ALO_2 = (AL_2, OL_2)$ is defined as follows:

$$\begin{aligned} &\text{Distance}(ALO_1, ALO_2) = \text{Distance}_{\text{AL}}(AL_1, AL_2) + \text{Distance}_{\text{OL}}(OL_1, OL_2) \\ &\text{With Distance}_{\text{AL}}(AL_1, AL_2) = \frac{1}{n} \cdot \sum_{i=1}^{i=n} \left| p_{1i} - p_{2i} \right| \text{ and} \\ &\text{Distance}_{\text{OL}}(OL_1, OL_2) = \frac{1}{N_c} \cdot \sum_{k=1}^{k=N_c} \left| OL_{1k} - OL_{2k} \right| \end{aligned} \tag{29}$$

where p_{1i} and p_{2i} are the position of activity i in AL_1 and AL_2, respectively.

The construction of $RefSet_1$ and $RefSet_2$ starts by sorting the new population *New_Pop* according to the lowest makespan. In case of tie, a second measure is used to choose the solution that overleap the least, similar to the second objective presented in section 6.3.1. The formulation adapted to the notations of the proposed SS is as follows:

$$\text{Overlap\_Measure(ALO)} = \frac{1}{2}\left(\frac{\sum_{(i,j)=A_q \in A} r_{ij,OL_q}}{\sum_{(i,j)\in A} r_{ij,m_{ij}}} + \frac{\sum OL_q}{\sum_{(i,j)\in A} m_{ij}} \right) \tag{30}$$

As a third measure to break the tie, we use a function used by Paraskevopoulos et al. (2012) and Berthaut et al. (2015) to measure the deviation of the schedule from the earliest finish times. The deviation is weighted with the activity durations and the resource consumptions. This function can be expressed as follows:

$$\begin{aligned} f(s) &= \sum_{i=1}^{i=n}\left(d_i * \frac{f_i - EF_i}{EF_i + 1} * \sum_{k=1}^{K} r_{ik} \right) \text{ for direct schedules,} \\ f(s) &= \sum_{i=1}^{i=n}\left(d_i * \frac{LF_i - f_i}{LF_i + 1} * \sum_{k=1}^{K} r_{ik} \right) \text{ for reverse schedules} \end{aligned} \tag{31}$$

$RefSet_1$ is initialized with the current best solution. The next best solutions of *New_Pop* are scanned and added to $RefSet_1$ if the minimal distance with the current members of $RefSet_1$ is superior or equal to t_1. In a second phase, the improvement algorithm of Berthaut et al. (2015) is applied to check if any solutions of *New_Pop* \ $RefSet_1$ can be added or can replace a member of $RefSet_1$ to increase the size of $RefSet_1$ (if inferior to b_1), or to improve the average makespan or the average distance between the members of $RefSet_1$. $RefSet_2$ is similarly built from the remaining solutions *New_Pop* \ $RefSet_1$.

Depending on the values of b_1, b_2, t_1 and t_2, there may not be enough qualified solutions to complete $RefSet_1$ and $RefSet_2$. If this situation arises when the refset update mechanism is applied on the initial population, we complete the reference sets with the best remaining solutions without checking the threshold conditions. Otherwise, a diversification strategy involving a frequency-based memory is used to complete the reference sets with new diversified solutions. We maintain for this purpose two matrix M_{AL} and M_{OL} that track the previously generated activity

and overlapping lists, respectively. Each element $M_{AL}(i,j)$ of matrix M_{AL} tracks the number of time activity i has been placed directly before activity j in previously generated activity lists. Similarly, each element $M_{OL}(q,m)$ of matrix M_{OL} tracks the number of time the couple $A_q \in A$ has been overlapped in mode m. A new solution represented with $ALO = (AL, OL)$ is generated as follows:

- AL is built step by step by adding at each stage k the activity j of the decision set D_k that has the least been placed directly after the activity AL_{k-1} at position (k - 1) in the list under construction (i.e., j such that $M_{AL}(AL_{k-1}, j) = \min_{r \in D_k}((AL_{k-1}, r))$).
- OL is build by assigning for each couple of overlapping activities the overlapping mode that has the least been used: $OL_q = \min_{m \in [1, m_{ij}], (i,j)=A_q} (M_{OL}(q,m))$.

6.4.7 Generate the subsets from Refsets

The subsets generation block forms the subsets of solutions that will be combined to create new solutions. We generate in *Subsets* all pairs of solutions in $RefSet_1$, and all combination of one solution from $RefSet_1$ and another one from $RefSet_2$. The total number of pairs to be combined is $\frac{b_1 \cdot (b_1 - 1)}{2} + b_1 \cdot b_2$.

6.4.8 Combination method - Path Relinking

PR is used as combination method to generate the next population *New_Pop* from the subsets of solutions (Marti et al., 2006). PR generates new solutions by exploring trajectories that connect high-quality solutions. In our SS algorithm, PR is applied on each pair in Subsets represented by ALO_1 and ALO_2 (such that the makespan of ALO_1 is better than the makespan of ALO_2), which constitutes the first guiding solution ALO_{guid} and the first initiating solution ALO_{init}, respectively. At each step of PR, a new solution ALO_{cu} is build from ALO_{init} and ALO_{guid}, and then added to the set of new solution *PR_set*, forming progressively the trajectory from ALO_1 to ALO_2. The proposed PR is a bidirectional PR that swaps ALO_{guid} and ALO_{init} at each step. This advanced type of PR was identified as a promising approach by Glover et al. (1995) and Marti et al. (2006) and applied to the RCPSP by Berthaut et al. (2015).

ALO_{cu} is build from ALO_{init} by making alternatively one move on the activity list AL_{init} and one move on the overlapping list OL_{init} towards ALO_{guid}. We introduce the following distance measure for each activity:

$$\text{Distance\_activity}(j) = \left| p_{init,j} - p_{guid,j} \right|, \forall \text{j} \in \{1, ..., \text{n}\} \tag{32}$$

where $p_{init,j}$ and $p_{guid,j}$ represent the position of activity j in AL_{init} and AL_{guid}, respectively.

The most distant activity z between AL_{init} and AL_{guid} is moved from its position in AL_{init} to its position in AL_{guid} to form the new AL_{cu}, and the remaining of the list is the same as in AL_{init}. If AL_{cu} is not precedence-feasible, a repair mechanism is invoked. Let p_{init} and p_{guid} be the position of z in AL_{init} and AL_{guid}, respectively. The repair mechanism works as follows:

1) If $p_{init} < p_{guid}$, then for forward (backward) schedules all successors (predecessors) of z between positions p_{init} and p_{guid} - 1 in AL_{cu} are moved right after z in the same order;

2) If $p_{init} > p_{guid}$, then for forward (backward) schedules all predecessors (successors) of z between position p_{guid} +1 and p_{init} in AL_{cu} are moved right before z in the same order.

For the overlapping list, we introduce a new move based on the following distance measure:

$$\text{Distance\_overlapping}(q) = \left| OL_{init,q} - OL_{guid,q} \right|, \forall q \in \{1, ..., N_c\} \tag{33}$$

where $OL_{init,q}$ and $OL_{guid,q}$ represent the mode of the couple OL_q in OL_{init} and OL_{guid}, respectively.

We set $OL_{cu} = OL_{init}$ and the most distant overlapping mode b between OL_{init} and OL_{guid} is changed by one unit towards the value in OL_{guid}. If $OL_{init,b} > OL_{guid,b}$, then $OL_{cu,b} = OL_{init,b} - 1$, else $OL_{cu,b} = OL_{init,b} + 1$.

The resulting $ALO_{cu} = (AL_{cu}, OL_{cu})$ is added to *PR_set*. We then set $ALO_{init} = ALO_{guid}$ and $ALO_{guid} = ALO_{cu}$ and repeat the process until $ALO_{cu} = ALO_{guid}$, thus connecting ALO_1 and ALO_2 with a single path composed of all the *ALO* stored in *PR_set*. Only a subset of *PR_set* will be selected for evaluation with the serial SGS. Let n_{pr} be the number of activity lists to be selected for evaluation. The activity lists in *PR_set* are numbered from 1 to |*PR_set*| and placed in n_{pr} subsets of size |*PR_set*| / n_{pr}. One *ALO* from each subset is randomly selected and evaluated to find its schedule and its makespan.

6.4.9 Improvement strategy - FBI

Search intensification is typically achieved in SS with the execution of an improvement method applied on the new solutions found during the combination process. Introduced by Li and Willis (1992) and popularized by Tormos and Lova (2001) and Valls et al. (2005), FBI iteratively applies forward and backward passes until no further improvement can be produced. In the forward (backward) pass, the activities are scheduled as early as possible (as late as possible). As this simple technique produces notable improvements with a small computation effort (Valls et al., 2005), it is introduced in the proposed SS.

In the first phase of the FBI, the backward pass is done by first applying the TO transformation from a forward (backward) solution to obtain a backward (forward) solution, by reversing the project network, and then by applying backward (forward) scheduling. In the second phase, the forward pass is applied in the same manner to obtain a forward (backward) schedule in the initial original (reversed) network direction. This process is repeated until the makespan can no further been improved. In the proposed SS, FBI is first applied on the best solutions of the initial population, and then only on high-quality solutions that match or improve the best makespan found so far during the process.

6.4.10 New Population generation

The new population *New_Pop* is first initialized with the best solution found so far. For each subset, n_{pr} solutions are generated with the combination method and possibly improved by FBI. This constitutes the new population for the next SS iteration, thus composed of $\left(\frac{b_1 \cdot (b_1 - 1)}{2} + b_1 \cdot b_2\right) \cdot n_{pr} + 1$ solutions.

6.5 Instance generator with overlapping

In order to evaluate and analyze the performances of the proposed exact and metaheuristic methods, we generated three sets of instances composed of projects with 30, 60 and 120 activities. The instances are derived from the PSPLIB benchmark designed for the standard RCPSP Kolisch and Sprecher (1997), and additional overlapping data are generated.

The PSPLIB benchmark was generated with the instance generator PROGEN (Kolisch et al., 1995), which is based on the characterization of projects according to the project size and three independent parameters *NC*, *RF*, and *RS*. *NC* corresponds to the average number of non-redundant precedence relations per activity, *RF* gives the average percent of different resource types required by an activity, and *RS* is a normalized indicator that defines how scarce the resources are. The instances for the extended problem with overlapping data are generated using a full factorial experimental design of these three parameters with one replication of each combination. The levels defined in Kolisch, Schwindt and Sprecher (1999) are used, namely $NC \in \{1.5,1.8,2.1\}$, $RF \in \{0.25,0.5,0.75,1\}$ and $RS \in \{0.2,0.5,0.7,1$. Therefore, 48 instances are considered for each set 30, 60 and 120 activities. For each instance, 27 instances with overlapping data are generated. Three additional parameters with three levels are introduced to generate the overlapping data: the proportion of couples of overlappable activities among all the precedence relations (*OC*), the rework rate (*RR*) and the maximum amount of overlap (*MO*). *OC* indicates how many precedence constraints can be relaxed to authorize overlapping, due to the nature of the project and the content of the activities. In order to characterize the rework with a single parameter *RR*, we assume a linear relation between the total amount of rework and the amount of overlap. Roughly speaking, *RR* represents the global sensitivity of the downstream activities to changes in the input information provided by the upstream activities. We also assume that the completion dates of the internal milestones of each upstream activity correspond to 0%, 25%, 50% and 75% of the duration of the activity. The maximum number of modes is 4. If *MO* = 75%, it means that the downstream activity could start once the upstream progress reaches the internal milestone associated with 1 - *MO* = 25%, and there are four modes (i.e., 0%, 25%, 50% and 75%). The maximum amount of overlap is also constrained by the durations of the activities in such a way that the downstream activity must start and end after the upstream activity. As the parameters of the projects are assumed to be integer values, the amounts of overlap and rework must be rounded. The overlapping data generation proceeds as follows for each PSPLIB instance:

1) Generation of the set *A* of couples of overlappable activities:
   a. The total number of couples of overlappable activities is given by $|A| = \text{round}\ (OC \cdot |E|)$,
   b. *A* is build by randomly selecting |*A*| couples of activities among the set *E*.

2) Generate the modes and the amounts of overlap for each couple $(i, j) \in A$:

The maximum overlap *MO* and the durations of the upstream and downstream activities d_i and d_j provide the amount of overlap associated to each mode as follows:

$$\alpha_{ij,m_{ij}} = \text{round}\left(d_i \cdot \min\left(\frac{d_j}{d_i}; MO\right)\right);\ m_{ij} = \text{ceil}\left(\frac{\alpha_{ij,m_{ij}}}{0.25}\right) + 1,$$

$$\alpha_{ijm} = \text{round}\ \left(0.25 \cdot (m-1)\right),\ \forall m \in \left\{2,..,m_{ij}-1\right\},$$

The modes with the same amount of overlap after rounding are merged into a single mode.

3) Generate the amounts of rework for each couple $(i, j) \in A$:

The amount of rework is computed by $r_{ijm} = \text{round}\ \left(\alpha_{ijm} \cdot RR\right),\ \forall m \in \left\{2,..,m_{ij}-1\right\}$.

The levels used to generate the whole benchmark set are summarized in Table 6.2. The benchmark set is a full factorial design of the seven parameters, thus composed of 3888 instances. This generation of overlapping data was introduced by Grèze et al. (2014b), but they used it to generate a smaller benchmark only composed of instances of 30 activities with constant values for *NC*, *RF* and *RR*.

Table 6.2: Parameter levels of the benchmark set

Size	***NC***	***RF***	***RC***	***OC***	***MO***	***RR***
30	1.5	0.25	0.2	0.2	0.25	0.25
60	1.8	0.5	0.5	0.4	0.5	0.5
120	2.1	0.75	0.7	0.6	0.75	0.75
		1	1			

6.6 Computational experiments

The exact procedure defined in section 6.3 is first used to find the optimal solutions of the 1296 project instances generated for 30 activities. The path relinking-based scatter search algorithm proposed in section 6.4 is then applied to the whole benchmark of 3888 project instances with 30,

60 and 120 activities. A statistical analysis is also presented to model the influence of the project characteristics on the makespan gain with overlapping.

6.6.1 Results with the exact procedure for 30 activities

The two-step exact procedure is first used to solve the RCPSP with overlapping for 30 activities. The upper bound T is initially set at M_{no} - 1, where M_{no} is the optimal makespan of the RCPSP without overlapping available on the PSPLIB website (http://www.om-db.wi.tum.de/psplib/files/j30opt.sm). The constraint propagation algorithm was implemented in Matlab R2011b, while ILOG-Cplex 12.5 was used as solver for the linear programming model. The tests were carried out on a personal computer with an AMD FX-6100 six-core clocked at 3.30 GHz with 16 GB RAM under Windows 7 Enterprise.

The average time to perform the constraint propagation algorithm is 0.39 sec. Infeasibility was detected for 136 out of the 1296 instances (10.49% of the benchmark set), which means that the optimal solution is to not overlap. For the 1160 remaining instances, an average reduction of 7.26% of the length of the time windows is observed. Within a time limit of 20000 sec., an optimal solution for the modified 0-1 integer linear programming model is found in 1105 instances (85.26% of the benchmark set).

The 55 remaining instances (4.24% of the benchmark set) that cannot be solved to optimality are mostly derived from instances known to be the hardest of PSPLIB. However, solving with 20000 sec. provides a lower bound LB of the makespan for each instance and possibly a best integer solution to be used as new upper bound T. Since the optimal solution is comprised between LB and T, a destructive procedure is introduced to improve the results. Destructive procedures consist in restricting the problem by setting a maximal objective function value F and trying to contradict (destruct) the feasibility of this reduced problem. In case of success, $F + 1$ is a valid new lower bound. These procedures have been found to be very effective for computing good lower bounds Klein and Scholl (1999). First, we impose a maximal value F for the makespan, comprised between LB and T. New time windows and new valid inequalities are derived with the constraint propagation algorithm. The linear programming model is then solved with a time limit of 20000 sec. If infeasibility is detected, then LB can be set to $F + 1$. If an integer solution is found, then T can be set at this integer solution. The process is repeated with a higher value of F and is stopped if an optimal solution is found, if $LB = T$, or if no integer solution or infeasibility is

found after several values of F. Note that if $LB = T$, then the optimal makespan is found but the optimal overlapping modes are not necessarily found. Out of the 55 instances tested with this destructive procedure, the optimal schedule and overlapping decisions have been found for 20 instances. For 8 other instances, an integer solution is found for the optimal makespan but the overlapping modes are not proved to be optimal. Finally, 27 instances cannot be solved (2.08% of the benchmark set), but the interval $\{LB, \ldots, T\}$ has been reduced.

The results are summarized in Table 6.3. Almost 98% of the instances have been solved to optimality with respect to the makespan, while almost 97% have been solved to optimality with respect to both the makespan and the overlapping decisions. The average CPU time required to solve the instances is 7184.81 sec. and about 70% of the instances can be solved in less than 10 sec. To measure the global effect of overlapping on the project duration, the makespans obtained for the RCPSP with and without overlapping are compared. The average makespan gain is 4.11% over the instances that are solved to optimality. If we take into account the final lower and upper bounds found of the 27 instances that cannot be solved, the average makespan gain over the whole subset of 30 activities belongs to the interval [4.04%, 4.11%].

Table 6.3: Results with the exact procedure for solving the RCPSP with overlapping

No. of instances	Prop. of instances	Optimal makespan reached	Optimal schedule known	Optimal overlapping modes known	Method
136	10.50%	yes	yes	yes	CP
1105	85.26%	yes	yes	yes	CP + LP
20	1.54%	yes	yes	yes	CP + LP, destructive procedure
8	0.62%	yes	yes	no	CP + LP, destructive procedure
27	2.08%	no	no	no	CP + LP, destructive procedure

6.6.2 Performances of the scatter search algorithm

As we observed an important computation effort for solving the RCPSP with overlapping for projects of 30 activities with an exact procedure, the need for heuristic procedures is all the more pertinent. The objective of this section is to show the efficiency of SS algorithm introduced in section 6.4 to tackle this issue. The algorithm is derived from the algorithm developed by

Berthaut et al. (2015) for the standard RCPSP. We present the comparison of the proposed SS algorithm with the exact procedure for the benchmark of projects with 30 activities.

The SS algorithm was implemented in Matlab R2011b. The experiments were conducted on a personal computer with an Intel Core I5 2.53 GHz processor and 4 GB RAM under Windows 7 Professional. As the proposed SS is based on random devices, each instance is solved ten times and the average results of the ten replications, as well as the 95% confidence interval, are provided. The algorithm is tested with 1000, 5000 and 50000 generated schedules as stopping criteria (Hartmann & Kolisch, 2000; Kolisch & Hartmann, 2006). Parameter tuning was performed by applying the local search process introduced in Berthaut et al. (2015). The results are presented in Table 6.4.

Table 6.4: Parameter Tuning for the proposed scatter search for the RCPSP with overlapping

Size	**30**			**60**			**120**		
Schedule limit	**1000**	**5000**	**50000**	**1000**	**5000**	**50000**	**1000**	**5000**	**50000**
InitPop	300	800	1000	200	600	1500	200	300	1300
b_1	11	22	75	8	15	38	6	11	17
b_2	6	12	24	4	8	22	4	7	16
t_1	0.6	0.5	0.5	0.9	1.3	1.4	1	1.8	1.8
t_2	1.5	1.4	1.5	1.6	1.8	2.1	2.3	2.7	3.2
n_{pr}	1	1	2	1	1	1	1	1	1

The performances of the metaheuristic are presented in Table 6.5. The row “Avg. dev. CPM” is the average deviation of the best makespan from the critical-path lower bound with overlapping. The rows “Avg. dev. LB” and “Avg. dev. UB” represent the average deviation from the best lower and upper bounds found in section 6.6.1, respectively. The row “Avg. dev. Opt.” is the average deviation from the optimal makespan for the 1261 instances solved to optimality in section 6.6.1. “Avg. No. of LB” and “Avg. Prop. of LB” are respectively the average number and the proportion of instances for which the best makespan reaches the best LB. “Avg. CPU”, “Max. CPU” and “Avg. No. sched.” represent the average and maximum computation times and the average number of generated schedules to reach the best solution, respectively. Table 6.5 shows that the average deviation of the best makespan from the best *LB* is 0.18% for the instances of 30 activities with a limit of 50000 schedules. The best *LB* is reached for a large proportion of the

instances (92%). If we only consider the 1261 instances solved to optimality in section 6.6.1, the average deviation of the best makespan from the optimal makespan is 0.10%. These results are obtained with an average computation time of 82.02 sec., far below the one observed with the exact procedure. In addition, the metaheuristic is able to reach the optimal makespan and the optimal overlapping decisions more quickly than the exact procedure for 58% of the instances. This shows that the proposed metaheuristic is very efficient for the RCPSP with overlapping. The method rapidly provides good quality solutions even for lower schedule limits. Also, high quality schedules and overlapping decisions are obtained for the hard instances for which no integer solution is found with the exact method.

Table 6.5: Detailed performances of the proposed Scatter search

Size	**30**			**60**			**120**		
Schedule limit	**1000**	**5000**	**50000**	**1000**	**5000**	**50000**	**1000**	**5000**	**50000**
Avg. dev. CPM	17.97%	17.53%	17.33%	14.61%	13.88%	13.31%	16.59%	15.69%	14.92%
Avg. dev. LB	0.65%	0.34%	0.18%	-	-	-	-	-	-
Avg. dev. UB	0.57%	0.26%	0.10%						
Avg. dev. Opt.	0.52%	0.25%	0.10%	-	-	-	-	-	-
Avg. No. of LB	959	1092	1191	-	-	-	-	-	-
Avg. Prop. of LB	74%	84%	92%	-	-	-	-	-	-
Avg. CPU (s)	2.12	7.06	82.02	4.86	17.68	142.04	35.18	43.74	273.02
Max. CPU (s)	10.85	57.44	1689.38	22.43	116.01	1583.15	47.70	300.43	2368.91
Avg. No. sched.	219	657	3268	259	930	5748	308	1083	6791

Table 6.6 compares the average makespan gain obtained with the two methods for 30 activities. For the SS algorithm, the 95% confidence interval with ten replications is considered. In addition, two overlapping measures are introduced for the overlap decisions: the proportion of overlapped couples among the set of couples of overlappable activities, and the average amount of overlap for the overlapped couples expressed as a proportion of the maximum possible amount of overlap. The overlapping measures for the two methods are quite similar, but overlapping appears to be slightly less used in the solutions found with the SS algorithm.

Table 6.6: Comparison of the exact procedure and the SS for the RCPSP with overlapping for 30 activities

	Makespan gain			Overlapping measures*	
	Interval (%) from Upper and Lower bounds	**Avg. gain (%)**	**95% Conf. Interval (%)**	**Avg. Proportion of overlapped couples (%)**	**Avg. amount of overlap (%)**
Exact procedure	[4.04, 4.11]			8.89	61.87
SS algorithm (50000 sched.)	-	3.96	[3.96, 3.96]	8.61	61.50

* for the subset of 1261 instances for which an optimal schedule is found with the exact procedure.

6.6.3 Efficiency of overlapping for 30, 60 and 120 activities

In this section, the efficiency of overlapping is analyzed by measuring the makespan gain obtained by the metaheuristic for the whole benchmark with 30, 60 and 120 activities. Ten repetitions are conducted for each instance in order to reduce the influence of the random devices. Table 6.7 presents the sensitivity of the makespan gain with respect to the stopping criterion. Not surprisingly, the makespan gain increases with the schedule limit. This can be explained by the fact that more effort is required to obtained good quality solution for the RCPSP with overlapping, which is harder to solve.

Table 6.7: Sensitivity of the makespan gain with respect to the stopping criterion

Size	**30**			**60**			**120**		
Schedule limit	**1000**	**5000**	**50000**	**1000**	**5000**	**50000**	**1000**	**5000**	**50000**
Avg. makespan gain	3.72%	3.82%	3.96%	4.44%	4.58%	4.68%	4.37%	4.51%	4.64%
95% confidence interval (%)	[3.69,3.74]	[3.82,3.83]	[3.96,3.96]	[4.42,4.47]	[4.56,4.60]	[4.67,4.69]	[4.35,4.40]	[4.49,4.53]	[4.63,4.65]

The histogram of the makespan gain for the whole benchmark set is presented in Figure 6.3 with the limit of 50000 schedules. The histogram shows that the makespan gain is widespread, with a minimum value of 0 and a maximum value of 28%. The overall mean is 4.43% and the standard

deviation is 4.68%. For 25.08% of the instances, overlapping does not yield to a makespan gain. For these instances, any overlapping decisions would be inefficient. This raises the question of the influence of each instance parameter on the makespan gain. This issue is tackled in the next section with a statistical analysis.

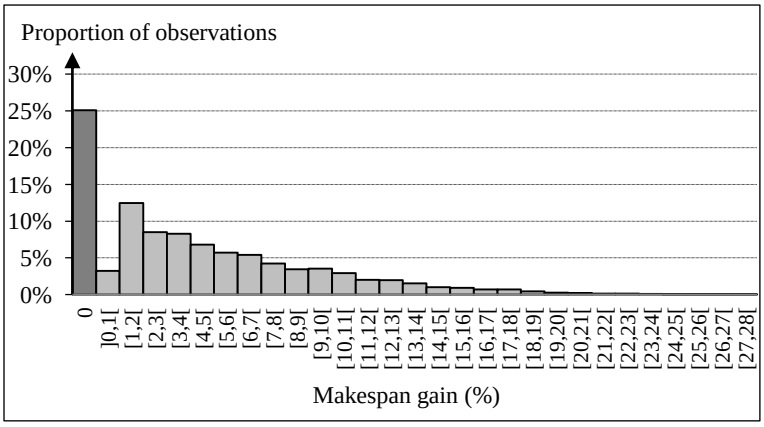

Figure 6.3: Histogram of the makespan gain with overlapping

6.6.4 Influence of the project characteristics on the efficiency of overlapping

Figure 6.3 highlights that the distribution of the makespan gain is characterized by a substantial mass point in 0 and a right skewed continuous distribution for the positive values. The use of typical linear regression would yield to inconsistent coefficients, and some negative values for the makespan gain. In presence of such so-called semi-continuous, which are prevalent in health and safety studies, economics, finance and insurance risk analysis (Yang & Simpson, 2010), two-part models can be used (Wooldridge, 2002). The first part models on when the outcome is null or positive, while the second part models on how much is the outcome conditional on its being positive. The estimation of the coefficients of the two models is provided by the maximum likelihood estimation method which can be factored into two independent terms so that the models can be estimated separately (Duan et al., 1983). The statistical analysis is accomplished using the statistical software application STATISTICA 12.

A logistic regression is used for the first part of the model. Let the variable *Gain* represents the makespan gain. The probability of the gain to be positive is modeled as follows:

$$\text{Prob}(Gain > 0 \mid (x_1,..,x_8, Rep)) = \frac{e^{g(x_1,..,x_8,Rep)}}{1+e^{g(x_1,..,x_8,Rep)}}$$
$$\text{with } g(x_1,..,x_8,Rep) = \beta_0 + \sum_{i=1}^{i=8}\beta_i \cdot x_i + \sum\sum_{i<j}\beta_{ij} \cdot x_i \cdot x_j + \sum_{i=1}^{i=8}\beta_{ii} \cdot {x_i}^2 + \sum_{k=1}^{k=10}\gamma_k \cdot Rep_k \quad (34)$$

where $(x_1,\ldots, x_8) = (Size, NC, RF, RS, OC, RR, MO, PC)$ represents the seven instance parameters and an additional continuous variable *PC* that measures the proportion of couple of overlappable activities on the critical path of the project without overlapping. The dummy variables Rep_k represent the effect of repeating on the results (i.e, $Rep_k = 1$ if $Rep = k$, 0 otherwise).

The second part of the model predicts how much the makespan gain is conditional on its being positive. Generalized linear models with gamma distribution and log link have been proposed in the literature of two-part models, because they are appropriate for strictly positive and right-skewed distribution. In addition, it is appealing because the variance of the observed makespan gain is heterogeneous, with a variance almost proportional to the square of the mean as assumed with the gamma distribution. The model is composed of the following elements:

- The gain is assumed to be generated from a gamma distribution, with a mean μ and a variance $V(\mu)$ proportional to the square of μ such that $V(\mu) = \mu^2 / \nu = \sigma^2 \cdot \mu^2$, where ν and σ represent the shape parameter and the coefficient of variation, respectively.
- The influence of the covariates is modeled by the linear predictor *f*:

$$f = \alpha_0 + \sum_{i=1}^{i=8}\alpha_i \cdot x_i + \sum\sum_{i<j}\alpha_{ij} \cdot x_i \cdot x_j + \sum_{i=1}^{i=8}\alpha_{ii} \cdot {x_i}^2 + \sum_{k=1}^{k=10}\delta_k \cdot Rep_k \quad (35)$$

- The log link function relates the linear predictor to the mean μ:

$$f = \ln(\mu) \quad (36)$$

The coefficients of the normalized variables for the two part of the model are presented in Table 6.8. Only the significant effects evaluated using the Wald's test with a threshold p-value of 0.05 are presented. For instance, repeating the metaheuristic procedure has a non-significant impact on the makespan gain.

Table 6.8: Results of fitting the logistic regression model and the gamma log link model

Effect	Logistic regression model			Gamma log link model (positive part)		
	Coefficient	Wald Statistic	p-value	Coefficient	Wald Statistic	p-value
Intercept	-5.582	1923.15	0.0000	-1.999	19082.73	0.0000
Size	-1.096	97.67	0.0000	0.543	1250.04	0.0000
NC	-0.279	129.91	0.0000	0.020	33.5	0.0000
RF	2.125	660.34	0.0000	-0.212	439.21	0.0000
RS	-3.925	1567.33	0.0000	0.820	5849.91	0.0000
OC	-0.625	76.73	0.0000	0.394	12715.39	0.0000
MO	-0.668	617.7	0.0000	0.315	8454.49	0.0000
RR	1.163	275.14	0.0000	-0.355	1563.25	0.0000
PC	-3.001	148.79	0.0000	1.151	1253.13	0.0000
*Size*NC*	0.155	38.76	0.0000	0.017	17.88	0.0000
*Size*RF*	-0.403	157.71	0.0000	0.088	284.85	0.0000
*NC*RF*	0.202	50.91	0.0000	—	—	—
*Size*RS*	-0.840	448.77	0.0000	0.109	361.56	0.0000
*NC*RS*	-0.155	24.43	0.0000	0.035	51.56	0.0000
*RF*RS*	-0.338	75.85	0.0000	0.264	2441.34	0.0000
*Size*OC*	0.254	71.38	0.0000	0.037	86.25	0.0000
*RF*OC*	0.268	80.15	0.0000	-0.071	255.96	0.0000
*RS*OC*	-0.488	188.06	0.0000	0.170	1217.25	0.0000
*Size*MO*	0.165	44.12	0.0000	0.026	46.26	0.0000
*NC*MO*	-0.070	8.06	0.0050	0.011	7.65	0.0060
*RF*MO*	0.209	53.44	0.0000	-0.021	23.67	0.0000
*RS*MO*	-0.358	109.1	0.0000	0.101	434.62	0.0000
*OC*MO*	-0.054	4.28	0.0390	0.022	30.17	0.0000
*Size*RR*	-0.158	30.78	0.0000	-0.022	22.72	0.0000
*NC*RR*	-0.149	34.47	0.0000	-0.019	22.41	0.0000
*RF*RR*	-0.162	29.87	0.0000	0.026	34.68	0.0000
*RS*RR*	0.406	136.84	0.0000	-0.056	134.16	0.0000
*OC*RR*	-0.291	131.6	0.0000	-0.040	97.98	0.0000
*MO*RR*	—	—	—	0.066	266.79	0.0000
*Size*PC*	-0.412	7.38	0.0070	0.555	503.1	0.0000
*RF*PC*	1.996	302.65	0.0000	-0.052	8.58	0.0030
*RS*PC*	-2.714	423.77	0.0000	0.411	491.95	0.0000
*OC*PC*	0.323	9.37	0.0020	—	—	—
*RR*PC*	0.496	26.12	0.0000	-0.046	8.41	0.0040
$Size^2$	0.320	55.24	0.0000	-0.046	39.31	0.0000
RS^2	0.536	163.8	0.0000	-0.419	5111.01	0.0000
OC^2	0.385	124.1	0.0000	-0.099	325.96	0.0000
MO^2	—	—	—	-0.127	531.27	0.0000
RR^2	0.141	16.1	0.0000	-0.091	277.47	0.0000
PC^2	1.861	144.82	0.0000	-0.363	155.64	0.0000

The ability of the logistic regression model to match the predicted and observed outcomes is measured with the Hosmer and Lemeshow's goodness of fit test with grouped data. As proposed by Paul, Pennell and Lemeshow (2013) for very large sample, the model was assessed on subsamples of approximately 1000 observations. The model shows no evidence of lack of fit (all p-value are above 0.05). With a cutpoint value of 0.5, the overall rate of correct classification between observed and predicted outcomes is 85.61%, with a rate of correct 1 (sensitivity) of 92.10% and a rate of correct 0 (specificity) of 66.23%. A more complete description of classification accuracy is given by the ROC curve (Receiver Operating Characteristic) which plots the sensitivity and 1 - specificity for the entire range of possible cutpoints. The area under the ROC curve provides a common measure of discrimination. This measure for the proposed logistic regression is 91.15%, which can be qualified as excellent according to Hosmer and Lemeshow (2000). Finally, the Nagelkerke's pseudo R-squared is 56.99%. It represents the proportion of explained variation on the log-likelihood scale and it measures the overall performance of the logistic regression. A logistic regression diagnostic was carried out to verify the adequacy of the model (Hosmer & Lemeshow, 2000).

The estimated coefficient of variation $\hat{\sigma} = 0.44$ for generalized linear model with gamma distribution and log link is derived from the method of moments. The modified Park Test confirmed that the gamma distribution is the most appropriate distribution to describe the variance-mean relation (Manning & Mullahy, 2001). If a more general link in the power family is introduced, the minimal deviance occurs with the null link parameter, which shows that the log link in also appropriate (McCullagh & Nelder, 1989). Finally, the overall performance of the generalized linear model is measured with a Nagelkerke's pseudo R-squared of 71.65%. The residual analysis based on the procedure of McCullagh and Nelder (1989) was carried out to verify the adequacy of the model.

Figure 6.4 and Figure 6.5 show the predicted probability of makespan gain and the predicted makespan gain (conditional on its being positive) as a function of the each independent variable, respectively. Each normalized variable is varied from -1 to 1 with all other variables held constant at their mean value. As expected, the probability of makespan gain and the predicted makespan gain (conditional on its being positive) increase when either the size of the project (*Size*), the average number of precedence relations per activity (*NC*), the resource capacity (*RS*),

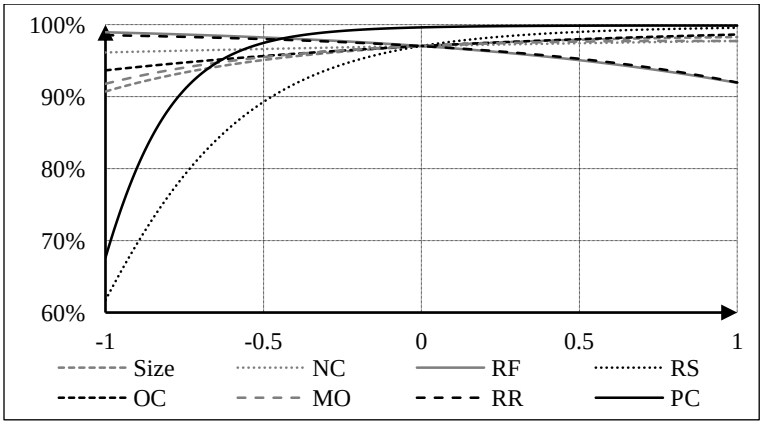

Figure 6.4: Predicted probability of makespan gain as a function of the parameters

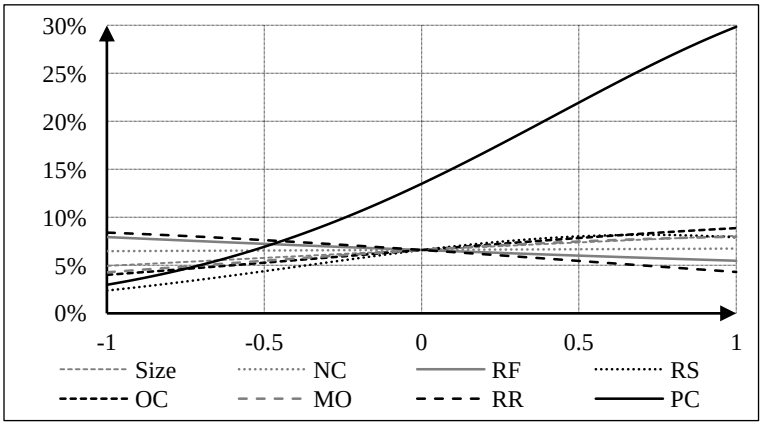

Figure 6.5: Predicted makespan gain (conditional on its being positive) as a function of the parameters

the proportion of couples of overlappable activities among all the precedence relations (*OC*), the maximum amount of overlap (*MO*), or the proportion of overlappable couples of activities on the critical path (*PC*) increases. It decreases when either the average number of different resource types required per activity (*RF*) or the rework rate (*RR*) increases. In addition, Figure 6.4 shows that both *PC* and *RS* have a higher impact on the probability of makespan gain, while the probability of gain is above 90% for any value of the other variables. Figure 6.5 highlights that *PC* has a higher influence on the makespan gain (conditional on its being positive) than the other parameters, while the *NC* has the lower influence. However, the makespan gain (conditional on its being positive) is below 10% when varying the parameters, with the exception of *PC*. This

means that the relative influence of *RS* on the makespan gain (conditional on its being positive) is more moderate than its influence on the probability of makespan gain.

As last step of the two-part model, the predicted makespan gain is obtained by combining the two models:

$$\begin{aligned} &\mathrm{E}\left(Gain \mid \left(x_1,..,x_8\right)\right) = \mathrm{Prob}\left(Gain > 0 \mid \left(x_1,..,x_8\right)\right) \cdot \mathrm{E}\left(Gain \mid \left(x_1,..,x_8\right), Gain > 0\right) \\ &\mathrm{E}\left(Gain \mid \left(x_1,..,x_8\right)\right) = \frac{e^{g\left(x_1,..,x_8\right)}}{1+e^{g\left(x_1,..,x_8\right)}} \cdot e^{f\left(x_1,..,x_8\right)} \end{aligned} \tag{37}$$

This two-part model applied to the whole set of instances exhibits a mean absolute error of 1.64% between the predicted and observed makespan gain and a coefficient of determination (R squared) of 74.96%.

6.6.5 The best overlapping decisions

The problem of selecting the solutions that overleap the least (see sections 6.3.1 and 6.4.6) gives rise to the following questions about the best overlapping decisions:

- What is the average proportion of overlapped couples among all the couples of overlappable activities?
- What is the average degree of overlapping per couple?
- In other words, should we overlap a lot of couples with a little degree of overlapping or should we overlap few couples with a large degree of overlapping?

Table 6.9 shows the frequency of distribution of the proportion of overlapped couples among all the couples of overlappable activities for the whole benchmark set. The table highlights that the proportion of overlapped couples is generally small, with a mean of 6.94%, a minimal value of 0% (i.e., no overlapping) and a maximal value of 45%. Almost 95% of the instances have a proportion of overlapped couples below 20%. If only the instances with an observed strictly positive makespan gain are considered, the mean proportion of overlapped couples is 9.23%. In addition, Table 6.10 highlights that the average amount of overlap among the overlapped couples (i.e., the ratio of the amount of overlap by the maximum possible amount of overlap) is quite large. The average value is 62.65%, with 0% as minimal value and 100% as maximum value. If

Table 6.9: Frequency of distribution of the proportion of overlapped couples

Proportion of overlapped couples among all the couples of overlappable activities	**0%**	**]0%,5%[**	**[5%,10%[**	**[10%,15%[**	**[15%,20%[**	**]20%,45%]**
Proportion of instances	26.09%	19.14%	25.22%	17.53%	6.97%	5.05%

Table 6.10: Frequency of distribution of the proportion of overlapped couples

Avg. amount of overlap	**0%**	**]0%,40%[**	**[40%,60%[**	**[60%,80%[**	**]80%,100%]**	**100%**
Proportion of instances	26.09%	1.31%	6.34%	18.18%	14.33%	33.75%

only the instances with a strictly positive makespan gain are considered, the mean value is 81.97% and more than 95% of the instances have an average amount of overlap above 40%. These results suggest that the best strategy would consist in overlapping only few couples of overlappable activities with a large degree of overlapping. This conclusion has several practical consequences for planning and controlling a project. First, effective overlapping can be applied by targeting only few couples of activities. Second, if overlapping requires specific communication and control processes in practice, this work only needs to be focused on few activities.

Overlapping should be limited to activities on the critical path of a project when no resource constraints are considered (Bogus et al., 2006; Khoueiry et al., 2013; Lim et al., 2014; Wang & Lin, 2009). The distribution of the observed proportion of overlapped couples on the critical path is given in Table 6.11. While the average proportion of couples of overlappable activities which are on the critical path without overlapping is 9.86% in the whole benchmark set, the average proportion of overlapped couples on the critical path among all the overlapped couples is 56.64%. This means that the couples of overlappable activities on the critical path are more likely to be overlapped. Also, for projects with a complex network and resource constraints, many couples of overlappable activities that are not critical are overlapped and thus the overlapping decision should not rely solely on the criticality of the activities.

Table 6.11: Frequency of distribution of the proportion of overlapped couples

Avg. proportion of overlapped couples which are on the critical path	**0%**	**]0%,40%[**	**[40%,60%[**	**[60%,80%[**	**]80%,100%]**	**100%**
Proportion of instances	31.25%	3.24%	10.25%	12.26%	4.73%	38.27%

6.7 Strategies for improving the efficiency of overlapping

Some authors have proposed practical strategies to improve the efficiency of overlapping based on the concepts of evolution of the upstream information and the sensitivity of the downstream activity introduced by Krishnan et al. (1997). In this section, we identify some of these strategies and we associate them with the project characteristics investigated in the present paper.

Bogus et al. (2006) identified several strategies for overlapping design activities in concurrent engineering and fast-tracking projects. Strategies to accelerate the evolution of the information so that the downstream activities can begin earlier include early freezing of design criteria, early release of preliminary information, prototyping, no iteration or optimization, and standardization of the design process and of the components. Other strategies, such as overdesign and set-based design, are proposed to reduce the sensitivity of the downstream activity. In addition, Wang and Lin (2009) recommended using advanced computer tools (e.g., CAD, CAM), better design practice (e.g., concurrent, engineering), lean product development and efficient team work in order to speed up the evolution of the information and to perform design changes more efficiently. They also proposed the use of Product lifecycle management systems and collaborative engineering tools to improve the communication between team workers.

Blacud et al. (2009) investigated the overlapping of design and construction activities in fast-tracking projects. This introduces the risk that physical rework has to be conducted on components under construction (e.g., changes in the foundation design while pouring of the concrete is in progress). They proposed practical strategies to reduce the sensitivity of the downstream activities to changes in the design: “delayed commitment strategy” (i.e., delaying the high transformation processes within a downstream construction activity), faster lead time alternatives for the components to be replaced, increased modularity of the system to be build, design of components with low sensitivity to design changes, use of modeling tools to analyze

construction activities and anticipate rework risks, and finally improved communication between design and construction teams.

In this paper, the strategies that allow downstream activities to begin earlier could be applied, on one hand, to increase the value of *OC* by identifying additional couples of overlappable activities, and , on the other hand, to increase the value of *MO* by allowing earlier transfer of information. In addition, these strategies can be specifically targeted towards the identification of additional overlapping couples on the critical path in order to increase the value of *PC*. In the literature, the amount of rework is assumed to depend on both the evolution of the information and the sensitivity of the downstream activity. Applying strategies to accelerate the evolution of the information and to reduce the sensitivity of the downstream activity could be useful to reduce the value of *RR*. Even though these strategies do not include any action concerning the resource constraints, the results in section 6.6.4 show that the scarcity of the resource has a major influence on the makespan gain. For this reason, we recommend the allocation of additional resource capacities (i.e., increase *RS*) in order to allow more tasks to be performed in parallel and to allow the execution of reworks. Table 6.12 summarizes these practical overlapping strategies and their relationship to the project parameters. The strategies are ranked according to their relationship to the most influent parameters observed in section 6.6.4. Note that we do not consider strategies for the parameters *Size*, *RF* and *NC*, because they can be considered as exogenous parameters.

Table 6.12: Summary of overlapping strategies to improve the efficiency of overlapping

Overlapping strategies	**Influence on *PC***	**Influence on *OC***	**Influence on *MO***	**Influence on *RR***	**Influence on *RS***
(1) Strategies to begin the downstream activities earlier (Bogus et al., 2006; Wang & Lin, 2009).	x	x	x	x	
(2) Strategies to add resource capacities.					x
(3) Strategies to reduce the sensitivity of downstream activities (Blacud et al., 2009; Bogus et al., 2006; Wang & Lin, 2009).				x	

6.8 Concluding remarks

Activity overlapping is one of the most employed strategies used to accelerate project execution. It entails that downstream activities start before the required information is available in a finalized form. However, overlapping often causes additional reworks in downstream activities that may outweigh the benefices of overlapping in terms of cost and time. The main contribution of the paper is to quantify and analyze the influence of eight project parameters on the reduction of the project makespan in projects with complex networks, resource constraints, overlapping and rework.

The reduction of the project makespan is obtained by solving the project scheduling problem with and without overlapping. Two methods have been developed for solving the problem with overlapping. First, we introduce a 0-1 integer linear programming model with overlapping modes and constraint propagation techniques as preprocessing. Second, we propose a metaheuristic based on a SS algorithm initially developed for the standard RCPSP. A set of 3888 project instances with overlapping is generated based on the PSPLIB library (Kolisch & Sprecher, 1997). The exact procedure is able to find the optimal makespan gain for 98% of the projects with 30 activities. Comparison of the two methods shows that the metaheuristic produces high-quality solutions in reasonable computational time. The metaheuristic method is subsequently applied on the set of projects of 60 and 120 activities.

The first finding is that no reduction of the makespan is observed in about 25% of the projects of the benchmark. Overlapping in these projects is not only useless, but it will also cause additional workload and costs. A statistical analysis is conducted to measure the effect of eight project parameters on the makespan gain obtained with the metaheuristic. The analysis is composed of a logistic regression to model on when the outcome is null or positive and a generalized linear model with gamma distribution and log link to model on how much is the outcome conditional on its being positive. This two-part model applied to the whole set of instances has a coefficient of determination (R squared) of 74.94%. All the project parameters and most of the interactions are significant. The magnitude of the coefficients reveals that the proportion of couples of overlappable activities on the critical path and the scarcity of the resource constraints have the highest influence on the makespan gain. The model could be used as a predictive model to evaluate the need for overlapping and to quantify the makespan reduction of a project.

Several authors have proposed practical overlapping strategies in the literature, but they did not discuss their relative efficiency on the makespan gain. By associating these strategies to the project parameters, we suggest that the strategies to add resource capacities and to begin the downstream activities earlier should prevail. Also, the best overlapping decisions should consist in overlapping only few couples of overlappable activities and to overlap them with a large degree of overlapping. Even though the activities on the critical path are more likely to be overlapped, overlapping decision should not rely solely on the criticality of the activities. A first extension of this study should be to examine other characteristics such as the consumption of resources in order to assist project managers and planners to choose the most appropriate activities to be overlapped.

A major limitation of our model is the estimation of the overlapping data. Some methods exist in the literature when organizations have experience with similar projects (Grèze et al., 2011; Grèze et al., 2014a; Krishnan et al., 1997; Lin, Chai, Brombacher, & Wong, 2009; Lin et al., 2010; Loch & Terwiesch, 1998; Roemer et al., 2000), but the problem of how to reliably estimate these data for new projects should also be investigated. In this regard, our predictive model could be used to evaluate the robustness of the makespan gain and the stability of the activity execution times in response to a non reliable estimation of the overlapping data. It could also be extended to the time-cost trade-offs between project duration and overlapping costs. Finally, the problem is formulated in a deterministic environment and does not directly address schedule risks. We may extend the model to introduce randomness, feedbacks and iterations. As some of the most advanced approaches in the literature for stochastic scheduling, such as proactive and reactive techniques, involve determining a baseline schedule without anticipation of uncertainty (Demeulemeester et al., 2008; Guéret & Jussien, 2008; Herroelen & Leus, 2004, 2005), the formulation in a deterministic environment proposed in this paper constitutes a first step towards the development of approaches for stochastic scheduling of projects with overlapping.

Acknowledgements

Financial support for this work was partly provided by the Natural Sciences and Engineering Research Council of Canada (NSERC). This support is gratefully acknowledged.

CHAPITRE 7 DISCUSSION GÉNÉRALE

Ce chapitre présente une discussion générale intégrée des aspects méthodologiques et des résultats des trois articles présentés dans cette thèse en lien avec les limitations de la littérature abordées dans le Chapitre 2, les autres travaux de recherche de la Chaire de recherche Jarislowsky / SNC-Lavalin en gestion de projets internationaux et l'applicabilité des modèles et des résultats à des projets industriels réels.

7.1 Contributions de la thèse par rapport à la littérature scientifique

Les limitations de la littérature scientifique sur l'ordonnancement de projets complexes avec chevauchement ont été présentées à la section 2.6 et peuvent se résumer de la façon suivante :

- Modélisation des durées de retouches en fonction des durées de chevauchement :
  Les travaux dans la littérature peuvent se classer dans les deux catégories suivantes :
  - modélisation détaillée du processus de chevauchement, sans considérer les contraintes de ressources et appliquée à des réseaux de projets simples ou de petite taille,
  - modélisation simplifiée du processus de chevauchement en posant comme hypothèse que la durée de retouche est proportionnelle à la durée de chevauchement ou qu'il n'y a aucune retouche, en considérant parfois les contraintes de ressources et appliquée à des réseaux de projet complexes.
- Prise en compte des contraintes de ressources :
  À l'exception de quelques rares travaux, les contraintes de ressources ne sont pas prises en compte. Pourtant, ces contraintes constituent une réalité des projets industriels et ont une influence sur les décisions de chevauchement. Également, elles constituent la principale cause de la difficulté à résoudre le problème d'ordonnancement de projets et conditionnent donc le choix de la méthode de résolution du problème d'ordonnancement.

- Méthodes de résolution du problème d'ordonnancement déterministe de projet avec chevauchement d'activités :

  À notre connaissance, il n'existe aucune méthode de résolution du problème d'ordonnancement déterministe de projet avec chevauchement d'activités autre que la méthode exacte de Gerk et Qassim (2008) basée sur la PLNE. Cependant, les limitations rencontrées par les méthodes pour le RCPSP classique sans chevauchement pour des projets de plus de 30 activités et avec des contraintes de ressources sévères indiquent que des méthodes de résolution approchées devraient être développées pour le problème d'ordonnancement déterministe de projet avec chevauchement d'activités pour aborder des problèmes de plus grande taille.

- Stratégies pratiques pour améliorer l'efficacité du chevauchement:

  - Les travaux sur la résolution du problème d'ordonnancement déterministe de projet avec chevauchement d'activités illustrent généralement leur approche à l'aide d'un seul exemple ou d'un nombre limité d'exemples. Ceci ne permet pas de dériver des enseignements ou des règles généraux sur les meilleures décisions de chevauchement, ni d'analyser l'effet de différentes caractéristiques de projets sur l'efficacité du chevauchement.
  - À l'inverse, certains auteurs ont proposé des stratégies pratiques pour augmenter la durée de chevauchement et limiter les retouches (section 2.5). Cependant, aucune analyse quantitative de l'effet de ces stratégies et de leur impact relatif les unes par rapport aux autres n'est proposée dans la littérature à notre connaissance.

Les contributions de cette thèse relative à la modélisation et à la résolution du problème d'ordonnancement de projets complexes avec chevauchement d'activités permettent de répondre en partie à ces limitations.

Premièrement, la modélisation de la durée de chevauchement en fonction de la durée de retouches basée sur les modes de chevauchement présentée dans la section 3.3.2, le Chapitre 4 et le Chapitre 6 autorise la modélisation de la durée de chevauchement autrement qu'avec une relation linéaire avec la durée de chevauchement. Cependant, la modélisation dans le Chapitre 4 permet de chevaucher uniquement à des instants précis représentant les jalons de l'activité en

aval, alors que la modélisation dans le Chapitre 6 propose que la durée de chevauchement soit une fonction constante par morceaux de la durée de chevauchement, dans laquelle chaque marche d'escalier correspond à un mode de chevauchement. En définissant un mode pour chaque unité de temps de l'intervalle possible de chevauchement, les deux modèles se rejoignent et permettent de représenter n'importe quelle relation entre durée de retouche et durée de chevauchement sous la forme d'une fonction discrète. De nombreux travaux sur la modélisation détaillée du chevauchement ont conclu que la durée totale des retouches est une fonction croissante et convexe du degré de chevauchement (section 2.4.3). À l'aide de notre modèle de chevauchement, il serait possible d'intégrer ce type de relation plus réaliste à l'ordonnancement des projets complexes composés de plusieurs couples d'activités chevauchables et avec contraintes de ressources.

La modélisation du problème d'ordonnancement de projet avec chevauchement et les méthodes de résolution proposéees dans le Chapitre 4 et le Chapitre 6 sont adaptées pour les contraintes de ressources et ont permis de confirmer la difficulté à résoudre ce problème en présence de contraintes de ressources. De plus, l'exemple illustratif présenté à la section 4.4, de même que l'analyse statistique effectuée à la section 6.6 prouvent et quantifient l'influence significative des contraintes de ressources sur les performances de projet et les décisions de chevauchement. En particulier, alors que de nombreux auteurs suggèrent de chevaucher les activités sur le chemin critique lorsqu'il n'y a pas de contraintes de ressource, nous montrons à la section 6.6.5 qu'en présence de contrainte de ressources les meilleures décisions de chevauchement ne consistent pas à se concentrer uniquement sur les activités critiques.

Enfin, l'application des méthodes de résolution exactes et approchées à un grand bassin d'instances virtuelles de projets dans le Chapitre 6 a permis de mieux cerner l'applicabilité de ces méthodes en fonction de la taille du problème d'ordonnancement. Alors qu'il est possible résoudre le problème classique RCPSP pour les 480 instances de projets de 30 activités avec une moyenne de temps de résolution de l'autre de 10 secondes (Artigues et al., 2013; Demeulemeester & Herroelen, 1997; Koné et al., 2011), l'ajout des décisions de chevauchement amène à une moyenne de temps de résolution de plus de 7000 secondes tel que présenté dans le Chapitre 6. La pertinence et la nécessité de développer des méthodes de résolution approchées pour le problème d'ordonnancement de projet avec chevauchement sont ainsi confirmées dans l'optique de résoudre des problèmes de taille plus importante. La métaheuristique développée

dans le Chapitre 5 et Chapitre 6 apporte une réponse à cet impératif en permettant d'obtenir des performances de qualité avec un temps de calcul raisonnable, en permettant de résoudre des projets de plus grande taille et en fournissant systématiquement une solution faisable.

Pour conclure, l'analyse des résultats obtenus sur le bassin complet d'instances virtuelles de 3888 projets de 30 à 120 activités dans les sections 6.6 et 6.7 a permis de quantifier l'influence de différentes caractéristiques de projet sur le gain en durée obtenu avec chevauchement, de proposer un modèle prédictif du gain en durée en fonction de caractéristiques de projet, d'identifier des règles générales pour les décisions de chevauchement, et de relier les stratégies pratiques de chevauchement proposées dans la littérature à notre analyse quantitative. Ceci apporte un changement de paradigme dans la littérature scientifique puisque cela permet d'assister les planificateurs de projet dans les décisions de chevauchement en général dans les situations où il est difficile d'estimer en pratique les durées de retouches.

7.2 Comparaison des résultats des différents chapitres

Les modèles étudiés dans le Chapitre 4 et le Chapitre 6 ont permis de modéliser et d'analyser différents facteurs du chevauchement :

- dans le Chapitre 4, l'influence de la durée de communication/coordination sur les performances et l'interaction entre les décisions de chevauchement, les contraintes de ressources, le coût et la durée de projet dans le Chapitre 4,
- dans le Chapitre 6, l'influence et les interactions de plusieurs caractéristiques de projets sur l'efficacité du chevauchement en termes de gain de durée de projet, et l'influence des méthodes de résolution sur les résultats.

Les modèles du Chapitre 4 et du Chapitre 6 se distinguent toutefois sur de nombreuses hypothèses, résumés dans le Tableau 3.1 de la section 3.3.3. Les distinctions concernent l'autorisation de débuter les activités en aval entre les jalons de l'activité en amont, la prise en compte des durées et coûts de coordination/communication, la considération des contraintes de coût et enfin le second objectif de chevaucher le moins possible parmi les solutions ayant la même performance. Ces chapitres se distinguent également dans le nombre d'instances de projets pour illustrer nos contributions, puisque le Chapitre 4 ne présente qu'un seul exemple à partir duquel il est difficile de tirer des conclusions générales. Ainsi, il est difficile de comparer les

résultats et les interprétations de ces deux chapitres. Toutefois, ils illustrent tous deux l'influence des contraintes de ressources sur le temps de résolution, sur les meilleures décisions de chevauchement ainsi que sur les performances de projet.

La comparaison du Chapitre 5 et du Chapitre 6 concerne les performances de la métaheuristique pour résoudre le problème d'ordonnancement de projets complexes sans et avec chevauchement d'activités. Pour ces deux problèmes, les résultats sur les bassins d'instances virtuelles de projet démontrent que la métaheuristique est très performante pour des projets de 30 activités :

- L'écart avec la durée optimale de projet avec une limite de 50000 solutions générées est ainsi nul pour le problème sans chevauchement et en moyenne inférieur à 0.18% pour le problème avec chevauchement;
- Le temps de résolution moyen avec une limite de 50000 solutions générées est largement inférieur à celui avec les méthodes exactes avec et sans chevauchement. Pour le problème sans chevauchement, nous obtenons un temps moyen de 3.50 secondes (Table 5.3) à comparer avec une dizaine de secondes avec les meilleures méthodes exactes de la littérature (Artigues et al., 2013; Demeulemeester & Herroelen, 1997; Koné et al., 2011). Pour le problème avec chevauchement, l'écart est encore plus significatif avec un temps moyen de l'ordre de 80 secondes avec la métaheuristique (Table 6.5) à comparer avec plus de 7000 secondes pour la méthode exacte. La métaheuristique garantit en outre de trouver des solutions faisables pour les instances de projets où la méthode exacte n'en trouve pas pour le problème avec chevauchement.
- La métaheuristique est capable de trouver également des solutions de haute qualité avec des temps de résolution petits. Ainsi, avec une limite plus petite de 1000 solutions générées, l'écart moyen avec la durée de projet optimale est de 0.10% et de moins de 0.65% sans et avec chevauchement, avec des temps de résolution respectivement de 0.27 seconde et 2.12 secondes.

Les résultats obtenus pour des instances de projets de 60 et 120 activités sont également très encourageants pour le problème avec et sans chevauchement. La métaheuristique se classe ainsi parmi les toutes meilleures méthodes approchées de la littérature pour le RCPSP sans chevauchement (Chapitre 5). Même si il n'existe pas de point de comparaison pour les résultats obtenus pour le problème avec chevauchement avec 60 et 120 activités, les résultats du Table 6.5

démontrent la capacité de la métaheuristique à obtenir assez rapidement (limite de 1000 solutions générées) des résultats très proches des résultats avec un temps de résolution très long (50000 solutions générées).

7.3 Comparaison des contributions avec les autres travaux de la Chaire Jarislowsky / SNC-Lavalin

Le problème du chevauchement dans l'ordonnancement de projets complexes constitue depuis les débuts de la Chaire Jarislowsky / SNC-Lavalin en gestion de projets internationaux en 2010 un axe majeur de recherche en collaboration avec notre partenaire industriel. Ceci a amené à une collaboration fructueuse entre les projets de recherche menés par de nombreux étudiants. Nous présentons dans cette section les liens avec ces travaux qui étendent la contribution de cette thèse.

Une variante du modèle présenté au Chapitre 4 en considérant que la durée de coordination/communication est négligeable dans le processus de chevauchement et en ne considérant que le problème de minimisation de la durée de projet avec chevauchement a été présentée à la conférence internationale IESM 2009 en collaboration avec l'étudiant à la maîtrise Lucas Grèze (Berthaut et al., 2011). Les travaux de maîtrise de Lucas Grèze présenté dans Grèze (2011) se sont ensuite concentré sur les contributions scientifiques suivantes : l'intégration de la résolution du problème d'ordonnancement avec chevauchement dans un processus de planification incluant l'évaluation des données de chevauchement nécessaire à l'application industrielle, et le développement d'une méthode heuristique. Un générateur de projets similaire à celui présenté dans cette thèse a également été développé pour générer un bassin d'instances virtuelles de projet afin d'analyser l'influence de certaines caractéristiques de projets sur les performances de projet avec chevauchement. Ce bassin est composé uniquement de projets de 30 activités et seul un sous-ensemble des caractéristiques de projets que nous avons analysé a été étudié. Cependant, ils ont analysé l'influence de ces caractéristiques sur les coûts de projet, ce qui n'est pas abordé dans cette thèse.

Le RCPSP classique et l'extension étudiée dans cette thèse que constitue l'ordonnancement de projets avec chevauchement d'activités suppose un découpage du projet et une estimation des paramètres à un niveau détaillé que constitue le niveau des activités. Pourtant, le processus de planification s'effectue dans la pratique souvent sous la forme d'une approche hiérarchique de

planification constituée de niveaux stratégiques, tactiques et opérationnels qui intègre à chaque niveau plus de détails au fur et à mesure que le projet est défini (Association for the Advancement of Cost Engineering, 2010; Cherkaoui, Pellerin, Baptiste, & Perrier, 2013; Herroelen, 2005; Knutson & Bitz, 1991). Dans un contexte hiérarchique, le problème de planification tactique « *Rough-Cut Capacity Planning* » (RCCP), qui consiste à ordonnancer les lots de travail et à estimer de manière agrégée les quantités requises de ressources, intervient plus tôt dans le processus de définition et de planification de projet que le RCPSP et avec un niveau de détail moindre. Les décisions obtenues avec la planification RCCP servent ensuite de contraintes à respecter dans l'ordonnancement opérationnel effectué avec le RCPSP. Dans ce contexte, l'étudiant à la maîtrise Georges Baydoun a étudié le problème RCCP avec chevauchements de lots de travaux en se basant sur le même modèle de chevauchement (Baydoun, 2014). Ses résultats montrent d'une part que l'ajout des décisions de chevauchement détériore le temps de résolution et que le chevauchement améliore aussi bien le coût de projet que sa durée.

Enfin, l'étudiante à la maîtrise Ariane Duchesne a étudié le problème du RCPSP stochastique dans la famille des approches d'ordonnancement proactives (section 2.2.2). L'objectif est de déterminer un calendrier de projet robuste dans un contexte de durées d'activité stochastique et de projets de grande taille. Il s'agit d'ajouter à un calendrier sans anticipation des incertitudes des tampons de temps entre les activités à l'aide d'une méthode heuristique. Le calendrier sans anticipation des incertitudes est obtenu dans sa démarche par la résolution du problème standard RCPSP à l'aide de la métaheuristique présentée dans le Chapitre 5 lorsque la solution optimale n'est pas connue.

7.4 Application industrielle de l'ordonnancement de projets complexes avec chevauchement d'activités

Ces travaux de recherche au doctorat ont été menés au sein de la Chaire Jaryslowsky / SNC-Lavalin en gestion de projets internationaux en collaboration avec un partenaire industriel. Cette collaboration étroite a pris la forme d'entrevues avec des experts en planification et en ingénierie, près de 70 heures de formation en interne, ainsi qu'un stage de 6 mois comme planificateur au chantier pour le projet Mont-Wright à Fermont. Ces rencontres ont permis de confirmer l'intérêt d'étudier l'impact du chevauchement sur l'exécution des projets. En effet,

l'accélération de projets de construction en « *fast-tracking* » ne constitue dans cette firme pas une alternative à un séquencement traditionnel sans chevauchement des phases d'ingénierie et de construction, mais bien le mode privilégié d'exécution. Mieux cerner les enjeux reliés aux décisions et aux risques de chevauchements, adapter les pratiques de planification et de suivi de projet et comprendre l'impact sur les performances de projet sont des objectifs pour diminuer la durée et les coûts de projet au bénéfice de notre collaborateur industriel et de ses clients.

7.4.1 Observations des pratiques en gestion de projets fast-tracking d'un partenaire industriel

Les projets d'ingénierie et de construction comportent des spécificités propres. Par exemple, la séquence de construction, qui va des travaux civils à l'installation des équipements ne correspond pas à la séquence de conception qui part à l'inverse des équipements majeurs et de la conception mécanique et électrique afin de dimensionner le reste du projet. Dans ce contexte, il est important que l'ingénierie soit suffisamment avancée, en particulier sur les équipements critiques du projet, avant que la construction ne débute, sinon des activités de construction doivent être retardées ou retouchées si des modifications ont lieu dans l'ingénierie. De même, il faut que les contrats d'achat des équipements majeurs soient octroyés au plus tôt de sorte de recevoir les données techniques des fournisseurs nécessaires pour finaliser les dessins d'ingénierie qui en dépendent, et afin de ne pas retarder les activités de construction. Par exemple, si un équipement majeur nécessite de reposer sur des fondations, il faut que le contrat d'achat ait été octroyé, puis que les données du fournisseur soient reçues puis intégrées aux dessins des fondations, tout ceci avant de débuter la de construction des fondations.

Le chevauchement entraîne également une augmentation de coûts indirects de gestion du projet. Prenons l'exemple de l'administration des contrats de construction. En effet, les activités de construction ne sont généralement pas menées directement par la firme de génie-conseil qui gère le projet, mais par des sous-traitants. Dans un projet en « *fast-tracking* », les appels d'offres contenant l'envergure des contrats à octroyer sont démarrés alors que l'ingénierie relative à cette envergure est encore en cours. Ainsi, il y aura de nombreuses modifications aux contrats à négocier lorsque l'ingénierie est peu avancée puisqu'il y aura parfois de nombreuses révisions techniques après l'octroi des contrats. Il y aura ainsi globalement plus d'avis de changements et de directives de chantier lorsqu'un projet est en chevauchement agressif. De plus, l'analyse des

soumissions des contractants et le choix du meilleur contractant sont rendus plus difficiles si ces soumissions sont basées sur une ingénierie partielle. Les ingénieurs résidents au chantier, dont le rôle principal est de contrôler la qualité des travaux et de coordonner la résolution des problèmes de conception entre les entrepreneurs au chantier et les équipes de conception qui ne sont pas au chantier, notent également une augmentation des questions des sous-traitants relatifs à des dessins incomplets. Lorsque le projet est accéléré de façon agressive, l'ingénierie est moins avancée au moment d'effectuer la construction et il y a le risque que plus d'incohérences de conception soient détectées au moment de faire la construction. Chacune des questions d'ingénierie nécessite d'être remontée à l'ingénierie de projets et un processus de vérification, modification et validation parfois long s'engage.

Nous avons aussi dégagé plusieurs remarques concernant l'ordonnancement de projet chez notre partenaire industriel. Premièrement, la planification et l'ordonnancement de projet ne tiennent pas tout le temps en compte directement les contraintes de ressources et le suivi du projet s'attarde alors principalement sur les activités critiques. Pour l'ingénierie, on examine si on a suffisamment de ressources humaines pour exécuter le calendrier de projet, et celui-ci peut être modifié si ce n'est pas le cas. Il s'agit donc d'un processus itératif qui ne tient pas compte directement des contraintes de ressources dans le problème d'ordonnancement. Il existe pourtant des contraintes de ressources importantes puisqu'il s'agit de ressources humaines hautement qualifiées, qu'il y a une contrainte sur la disponibilité des bureaux et qu'il est impossible d'ajouter des quarts de travail supplémentaire. Pour la construction, les activités de construction sont généralement sous-traitées à des entrepreneurs qui effectuent leur gestion des ressources humaines. Il y a plus de flexibilité en construction pour augmenter le nombre de travailleurs ou ajouter des quarts de travail. Cependant les contraintes de ressources en construction existent souvent sous la forme de contraintes d'espace pour éviter l'encombrement et prendre en compte les consignes de sécurité. Afin de faciliter la gestion de ces contraintes, les corps de métier ne travaillent généralement pas au même moment à un endroit donné.

Dans la pratique de planification et d'ordonnancement, de nombreuses stratégies pratiques telles que celles décrites dans la section 2.3 sont déjà utilisées à différents degrés dans les projets de construction en « *fast-tracking* » chez notre partenaire industriel. Par exemple, l'utilisation d'outils collaboratifs et de système d'information facilite la communication et la coordination dans un contexte de retouches et de changements d'ingénierie et de construction. Également, les

équipes de constructions sont souvent impliquées assez tôt dans l'ingénierie afin de s'assurer qu'une analyse de constructibilité ait été effectuée et de coordonner l'ingénierie et la construction.

L'estimation des durées d'activités dans le calendrier de projets ne prend pas en compte les durées possibles de retouche. La principale raison est que pour des projets composés parfois de plusieurs milliers d'activités, il est difficile de récolter des données fiables à ce sujet. Un certain équilibre entre la granularité des détails, la fiabilité des informations et le temps nécessaire pour les estimer doit être fait. L'analyse du chevauchement devrait prendre en compte le processus de planification en pratique notamment au niveau du découpage du projet et des données d'estimation et de suivi des durées et coûts associés. Ainsi, l'estimation de ces données et le recueil des informations lors du suivi de l'exécution ne sont font pas nécessairement au niveau de détail des activités, mais parfois au niveau supérieur des lots de travaux, car un projet d'envergure peut comporter des milliers d'activités. Il est également compliqué d'examiner le chevauchement au niveau des lots de travail puisque chez notre partenaire industriel le découpage de l'ingénierie par lot de travail n'a pas de correspondance systématique avec le découpage de la construction qui s'effectue par contrats. Le chevauchement pourrait être examiné à des niveaux supérieurs, par exemple entre les disciplines (génie civil, mécanique, électrique,…), ou globalement entre l'ingénierie, l'approvisionnement et la construction. Par exemple, quel doit être l'avancement global de l'ingénierie et de la définition du projet quand débute la construction ?

7.4.2 Comparaison avec les modèles et résultats de la thèse

Les observations présentées dans la section précédente montrent que la plupart des projets de notre collaborateur industriel sont effectués en « *fast-tracking* » et que de nombreuses stratégies sont mises en place pour faciliter le chevauchement et la coordination entre les parties prenantes. Il n'y a par contre actuellement pas de méthodologie spécifique à l'ordonnancement de projets pour proposer des calendriers de projet prenant en compte le chevauchement, les contraintes de ressources conjointement, et différents objectifs du projet (durée et coût). Ceci confirme l'intérêt de nos objectifs de recherche.

Pour des projets de taille modeste et si l'organisation possède une expérience sur des projets similaires, il peut être possible d'estimer les durées de retouches en fonction des durées de

chevauchement au niveau des activités. Il s'agit du point de vue présenté dans les travaux de la littérature qui propose des modèles d'ordonnancement de projet avec chevauchement d'activités. C'est également l'hypothèse que nous avons prise dans le Chapitre 4.

Plusieurs travaux ont analysé la taille des projets en pratique et ont noté que presque la moitié des projets ont moins de cent activités (Icmeli-Tukel & Rom, 1997; Liberatore & Pollack-Johnson, 2003; Liberatore, Pollack-Johnson, & Smith, 2001). Le développement d'une méthode de résolution approchée dans la famille des métaheuristiques dans le Chapitre 6 permet d'étendre l'application de telles approches à des projets jusque 120 activités et donc de répondre aux besoins de beaucoup de projets dans la pratique.

Cependant, ces mêmes études ont montré que de nombreux projets sont de très grande taille. Selon ces auteurs, la proportion de projets de 1000 activités est de 12 à 14%. Cette proportion atteint même 19% pour les projets de construction selon Liberatore et al. (2001) et Liberatore et Pollack-Johnson (2003). Dans le cas de projets de très grande taille, l'approche qui consiste à estimer les données de chevauchement pour chaque couple d'activités chevauchables paraît impraticable. Ceci est renforcé par le manque de fiabilité pour estimer ces données si le projet est nouveau et qu'il n'y a pas de données historiques disponibles. L'exploitation des données historiques lorsqu'elles existent au niveau des activités peut également être compliquée par la difficulté à distinguer dans l'allongement de la durée des activités ce qui est causé par le chevauchement par rapport à d'autres causes (mauvaise estimation initiale, ajout d'envergure au projet, …). L'analyse statistique et le modèle proposés dans le Chapitre 6 peuvent dans cette optique être utilisés pour analyser l'impact de la fiabilité des données de chevauchement puisque le modèle permet d'évaluer l'impact de la variation des paramètres de projets.

Les résultats que nous avons trouvés dans le Chapitre 6 procurent aussi une aide à la décision aux planificateurs de projets dans le cas de tels projets de grandes tailles. En effet, l'analyse statistique de l'efficacité du chevauchement, le modèle prédictif du gain en durée en fonction de caractéristiques générales de projet, l'analyse des meilleures décisions de chevauchement et la comparaison avec les stratégies pratiques proposées dans la littérature nous ont permis de proposer des règles générales qui ne nécessitent pas l'estimation de toutes les données de chevauchement.

7.4.3 Hypothèses limitant l'application à des projets industriels

Le modèle de chevauchement et le modèle d'ordonnancement de projet étudiés dans ce projet comportent des hypothèses de modélisation qui limitent leur application à des projets réels tels que ceux de notre partenaire industriel :

- L'analyse de sensibilité et les règles générales relatives aux meilleures décisions de chevauchement présentées dans le Chapitre 6 supposent que les caractéristiques de projet sont les mêmes à l'échelle du projet. Cependant, les contraintes de ressources peuvent varier dans le temps, alors que les taux de retouches et la durée maximale de chevauchement peuvent être différents selon la nature des activités chevauchables;
- Notre analyse est faite dans un contexte déterministe et ne prend pas en compte les incertitudes, si ce n'est de manière indirecte dans la durée estimée des retouches. Terwiesch et Loch (1999) ont pourtant noté que le chevauchement est moins efficace pour diminuer la durée de projet lorsque les incertitudes sont importantes. Également, les itérations entre activités sont ignorées dans notre modèle;
- Les contraintes de ressource dans la pratique peuvent être plus flexibles. Il est possible de faire des heures supplémentaires, affecter des ressources supplémentaires ponctuellement, effectuer des quarts de travail supplémentaire. L'utilisation de tels moyens à cependant une influence sur le coût des activités.
- Nous avons considéré uniquement des relations de précédence de type fin-début pour les activités qui ne peuvent pas être chevauchées.

La relaxation de certaines hypothèses et une meilleure modélisation du problème d'ordonnancement figurent parmi les voies de recherche futures présentées dans la conclusion.

7.5 Conclusion

Ce chapitre a permis de mettre en perspective les aspects méthodologiques, les résultats et les contributions de cette thèse avec la littérature scientifique et les autres travaux de la Chaire de recherche Jarislowsky / SNC-Lavalin en gestion de projets internationaux. Ce chapitre a aussi permis de discuter de l'applicabilité à des projets industriels et de dégager quelques limitations dont la relaxation constitue une des avenues de recherche futures abordées dans la conclusion.

CHAPITRE 8 CONCLUSION ET RECOMMANDATIONS

Les objectifs de cette thèse sont de modéliser l'impact des décisions de chevauchement d'activités sur les performances de projet dans un contexte déterministe, d'analyser l'effet des caractéristiques de projet sur l'efficacité du chevauchement pour accélérer l'exécution d'un projet et enfin de proposer des stratégies générales applicables dans la pratique de la planification des projets complexes.

Le chevauchement d'activités est en effet une des pratiques les plus répandues pour accélérer l'exécution des projets industriels. Les stratégies d'ingénierie simultanée pour le développement de produits et les projets de construction en « *fast-tracking* » sont basées sur cette technique. Alors que les gestionnaires de projet sont confrontés à la question de quand et combien chevaucher, ils ne possèdent pas d'outils d'aide à la décision opérationnelle pour déterminer un calendrier de projet avec chevauchement et se basent principalement sur leur expérience pour ces décisions.

Cette problématique, reconnue par la communauté scientifique, a amené au développement de nombreux modèles détaillés de chevauchement et de nombreux modèles d'ordonnancement de projets avec chevauchement. La littérature existante possède cependant plusieurs lacunes qui portent principalement sur la prise en compte des contraintes de ressource, la modélisation du chevauchement des couples d'activités chevauchables et les méthodes de résolution du problème de détermination d'un calendrier de projet avec chevauchement. Ces lacunes limitent l'application de ces contributions à la pratique de la planification de projets industriels, qui se caractérisent notamment par des contraintes sur la disponibilité des ressources spécialisées, la complexité des relations de dépendance entre activités et un nombre important d'activités.

Nous présentons dans les sections suivantes un rappel des objectifs de cette thèse, une synthèse des travaux réalisés dans cette thèse pour les réaliser, les contributions scientifiques et pratiques, et les limitations de nos travaux, puis nous identifions plusieurs avenues de recherche futures.

8.1 Objectifs de recherche

Les objectifs de cette thèse présentée dans la section 3.1 ont été subdivisés en quatre sous-objectifs :

- Proposer un modèle réaliste de chevauchement d'activités qui puisse être intégré au problème d'ordonnancement de projet avec chevauchement;
- Développer des méthodes de résolution exactes et approchées pour le problème d'ordonnancement de projet avec chevauchement;
- Quantifier l'impact des décisions de chevauchement d'activités sur l'efficacité du chevauchement en termes de durée de projet en fonction de certaines caractéristiques de projets;
- Analyser les décisions de chevauchement pour apporter une meilleure compréhension de ces choix dans les projets complexes et proposer des stratégies générales applicables en pratique.

Nous présentons dans les deux sections suivantes une synthèse de travaux et des contributions de cette thèse qui ont amené à la réalisation de ces objectifs de recherche.

8.2 Synthèse des travaux

Les travaux de cette thèse ont couvert plusieurs aspects du problème conjoint d'ordonnancement de projet et de détermination des décisions de chevauchement. Premièrement, nous avons développé un modèle de chevauchement de deux activités réalistes et applicables pour le problème d'ordonnancement de projet avec plusieurs couples d'activités chevauchables. Nous l'avons ensuite intégré à la modélisation du problème d'ordonnancement de projet avec chevauchement d'activités. Nous avons proposé une méthode de résolution exacte basée sur la PLNE pour le problème de compromis coût-durée avec durée et coût non négligeables de coordination/communication et de retouches. Nous avons également proposé une méthode de résolution exacte basée sur la PLNE et sur la PPC et une métaheuristique basée sur la recherche dispersée pour le problème de minimisation de la durée de projet avec durées négligeables de coordination/communication. Nous avons au préalable démontré que cette métaheuristique de type recherche dispersée (« *scatter search* ») est parmi les méthodes approchées les plus performantes de la littérature scientifique. L'analyse des résultats a permis de dériver un modèle statistique qui quantifie l'effet du chevauchement en fonction de caractéristiques de projet et de déterminer les caractéristiques qui ont le plus d'impact. L'analyse des résultats a permis aussi de

dériver des recommandations générales et de confronter nos résultats aux stratégies pratiques proposées dans la littérature pour favoriser le chevauchement.

Ces travaux sont articulés autour de trois chapitres qui font chacun l'objet d'un article scientifique publié ou soumis dans des revues scientifiques internationales avec comité de lecture. Les trois chapitres peuvent se résumer de la façon suivante :

- Chapitre 4 : le premier article intitulé « *Time-cost trade-offs in resource-constrained project scheduling problems with overlapping modes* » (publié en 2014 dans International Journal of Project Organisation and Management) introduit un modèle de chevauchement d'activités basé sur des modes de chevauchement reliés aux jalons internes des activités. Ce modèle est inséré dans une modélisation du problème de compromis durée-coût de l'ordonnancement de projets complexes avec contraintes de ressource et chevauchement d'activités sous la forme d'un programme linéaire en nombres entiers avec modes d'activité. Les durées de communication/coordination et de retouches sont considérées. Le modèle est résolu avec une méthode exacte pour un exemple virtuel de projet. Les résultats illustrent les interactions entre le coût de projet, la durée du projet et les contraintes de ressource et leur influence sur le temps de résolution.

- Chapitre 5 : le second article intitulé « *A Path Relinking-based Scatter Search for the Resource-Constrained Project Scheduling Problem* » (soumis dans European Journal of Operational Research) introduit une métaheuristique dans la famille des recherches dispersées (« *scatter search* ») pour résoudre le problème standard RCPSP sans chevauchement. Cet algorithme utilise la méthode FBI, inverse la direction d'ordonnancement à chaque itération et est basé sur deux mécanismes novateurs. Premièrement, un PR bidirectionnel avec un nouveau mouvement opérant sur les distances entre activités est utilisé comme méthode de combinaison de solutions. Deuxièmement, une méthode d'amélioration est utilisée pour améliorer la qualité et la diversité des solutions de l'ensemble de référence. Une méthode avancée de paramétrage des paramètres de l'algorithme utilisant une méthode de recherche locale a été developpée pour déterminer les meilleures valeurs de ces paramètres. L'article montre que cette métaheuristique est capable de fournir des solutions de grande qualité avec des temps de

calcul acceptables et appartient aux meilleures méthodes approchées existantes dans la littérature pour la résolution des instances virtuelles de projet de PSPLIB.

- Chapitre 6 : le troisième article intitulé « *Influence of the project characteristics on the efficiency of activity overlapping* » (soumis dans Computers & Operations Research) a pour principales contributions de quantifier et d'analyser l'influence de huit caractéristiques de projets sur l'efficacité du chevauchement pour diminuer la durée de projet. La réduction de la durée de projet est obtenue en résolvant le problème RCPSP avec et sans chevauchement. Deux méthodes de résolution sont développées pour résoudre le problème avec chevauchement. Une nouvelle méthode exacte basée sur un programme linéaire en nombres entiers avec modes de chevauchement et des techniques de propagation de contraintes est développée. La seconde méthode est une métaheuristique dérivée de la métaheuristique proposée dans le second article. Ces méthodes sont appliquées à un bassin d'instances de projets générés avec chevauchement. Une analyse statistique permet de distinguer l'influence des paramètres de projets sur l'efficacité du chevauchement. Également, les résultats indiquent que certains principes généraux se dégagent pour les décisions de chevauchement. Enfin, ces observations sont confrontées aux stratégies pratiques de chevauchement proposées dans la littérature.

8.3 Principales contributions à la recherche

Les principales contributions de cette thèse en comparaison à la littérature scientifique existante et la pratique de la planification de projets industriels ont été présentées en détail dans le chapitre précédent. Elles sont relatives à la modélisation du chevauchement de deux activités, à la modélisation du problème d'ordonnancement de projets complexes avec chevauchement d'activités, à la résolution des problèmes d'ordonnancement de projets et à l'applicabilité de ces méthodes à la planification des projets industriels avec chevauchement. Ces contributions peuvent être résumées ainsi :

- Modélisation plus réaliste de la durée de chevauchement en fonction de la durée de retouches basée sur les modes de chevauchement et les jalons internes des activités. Cette modélisation présentée dans la section 3.3.2, le Chapitre 4 et le Chapitre 6 autorise la modélisation de la durée de chevauchement autrement qu'avec une relation linéaire avec

la durée de chevauchement, tout en pouvant être inséré dans un modèle d'ordonnancement de projet complexe comportant plusieurs couples d'activités chevauchables;

- Modélisation plus réaliste du problème d'ordonnancement de projet avec chevauchement d'activités. Les modèles du Chapitre 4 et du Chapitre 6 intègrent en effet les contraintes de ressources et le modèle de chevauchement d'activités avec modes de chevauchement pour l'ordonnancement de projets possédant un réseau complexe d'activités;
- Méthode de résolution performante pour le problème classique RCPSP. La métaheuristique de type recherche dispersée (« *scatter search* ») proposée dans le Chapitre 5 intègre des mécanismes novateurs et figure parmi les meilleures méthodes approchées existantes dans la littérature. De plus, cet algorithme est flexible pour être utilisé pour des extensions du problème RCPSP tel que l'ordonnancement de projet avec chevauchement d'activités;
- Méthode de résolution exacte et approchée performante pour le problème d'ordonnancement de projet avec chevauchement d'activités. La modélisation du problème avec la PLNE et utilisant la PPC comme prétraitement permet de résoudre près de 98% des problèmes de 30 activités dérivées de PSPLIB (Chapitre 6). La capacité des méthodes exactes à résoudre des problèmes d'ordonnancement pour des projets de grande taille étant limitée, cette thèse présente à notre connaissance la première méthode approchée de type métaheuristique pour ce problème (Chapitre 6). La comparaison des résultats de cette méthode avec ceux de la méthode exacte montre que pour des projets de 30 activités, la méthode approchée fournit des solutions de haute qualité avec des temps moyens de calcul réduits;
- Introduction d'un second objectif pour choisir la solution qui chevauche le moins parmi les solutions ayant la même durée de projet. Ce second objectif utilisé dans la méthode de résolution exacte et la métaheuristique au Chapitre 6 permet de déterminer non seulement le calendrier de projet ayant la durée de projet la plus petite, mais également celle qui utilise le moins le chevauchement. Ce second objectif n'est pas utilisé dans la littérature scientifique à notre connaissance et a permis dans un second temps d'analyser les décisions de chevauchement les plus efficaces pour réduire la durée de projet;

- Quantification et analyse de l'effet de huit caractéristiques de projet sur l'efficacité du chevauchement d'activités pour diminuer la durée d'un projet. Ces huit caractéristiques représentant la complexité du réseau d'activités, les contraintes de ressource et le chevauchement ont toutes un impact significatif sur l'efficacité du chevauchement d'activités pour diminuer la durée d'un projet (Chapitre 6). En particulier, la proportion de couples d'activités qui sont sur le chemin critique et la sévérité des contraintes de ressources ont un impact plus important sur l'efficacité du chevauchement;
- Le modèle statistique proposé peut être utilisé par les praticiens comme modèle prédictif dans le cas de projets de taille petite ou moyenne pour quantifier a priori l'effet du chevauchement sur la diminution de la durée de projet en fonction des caractéristiques de projet;
- Proposer des stratégies pratiques de décisions de chevauchement lorsqu'il n'est pas possible en pratique de déterminer les données de chevauchement. L'analyse des décisions de chevauchement permettant la meilleure diminution de la durée de projet montre dans le Chapitre 6 que la meilleure stratégie devrait consister à chevaucher peu de couples d'activités chevauchables et de les chevaucher beaucoup. De plus, même si les activités sur le chemin critique ont plus de probabilités d'être chevauchées, les décisions de chevauchement dans un contexte de contraintes de ressource ne doivent pas uniquement être basées sur la criticalité des activités;
- Relier les stratégies pratiques de chevauchement dans la littérature au modèle prédictif de l'efficacité du chevauchement. De nombreux auteurs ont proposé dans la littérature des stratégies pratiques de chevauchement, mais n'ont pas discuté dans leur efficacité relative pour diminuer la durée de projet. En associant ces stratégies au modèle prédictif présenté au Chapitre 6, nous suggérons que les stratégies d'ajout de capacités de ressources et des stratégies pratiques pour commencer les activités en aval plus tôt devraient être privilégiées. Ces dernières inclus le gel au plus tôt des critères de conception, la communication au plus tôt des informations préliminaires de l'activité en amont vers l'activité en aval, le prototypage, la limitation des itérations et de l'optimisation de la conception au sein de l'activité en amont, la standardisation du processus de conception et des composants, l'utilisation d'outils informatiques avancés (CAO, FAO, IAO,…),

l'application des principes du Lean Product Development (LPD), l'utilisation d'outils de Product Lifecycle Management **(**PLM) et d'ingénierie collaborative, le développement d'architecture de produits modulaires.

8.4 Principales limitations des travaux

Plusieurs limitations des travaux de cette thèse ont été identifiées dans les conclusions du Chapitre 4, du Chapitre 5 et du Chapitre 6 ainsi que dans le chapitre précédent. Nous rappelons ici les principales limitations :

- L'applicabilité de modèle d'ordonnancement de projet avec chevauchement d'activités pour un projet donné requiert l'estimation des données de chevauchement et de communication/coordination. Si plusieurs démarches ont été proposées dans la littérature pour cela lorsque les organisations ont une expérience avec des projets similaires (Grèze, Pellerin, & Leclaire, 2011; Grèze, Pellerin, Leclaire, & Perrier, 2014a; Krishnan et al., 1997; Lin, Chai, Brombacher, & Wong, 2009; Lin et al., 2010; Loch & Terwiesch, 1998; Roemer et al., 2000), le problème d'estimer de façon fiable ces données pour des projets nouveaux reste entier. Toutefois, l'analyse sur de nombreuses instances virtuelles de projet (Chapitre 6) permet dans une certaine mesure de s'affranchir de ces contraintes en proposant des recommandations générales et un modèle prédictif basé sur des caractéristiques générales de projet;
- Le problème d'ordonnancement de projets avec chevauchement d'activités est formulé dans un contexte déterministe. Cette thèse n'aborde pas directement les risques de dépassement des coûts et de la durée de projet alors que le chevauchement est intrinsèquement risqué puisqu'il suppose de se baser sur des informations préliminaires. Terwiesch et Loch (1999) ont ainsi montré que le chevauchement permet de réduire la durée d'exécution des projets, mais que son efficacité dépend grandement de l'incertitude du projet;
- Les itérations d'activités ne sont pas prises en compte. Nous supposons dans cette thèse que le flux d'information entre les activités est unidirectionnel et que les rétro-informations ont été éliminées à l'aide de technique utilisant les DSM (section 3.3.1). Cependant, ce n'est pas toujours possible en pratique;

- L'analyse de sensibilité et les règles générales relatives aux meilleures décisions de chevauchement présentées dans le Chapitre 6 supposent que les caractéristiques de projet sont les mêmes à l'échelle du projet. Cependant, les contraintes de ressources peuvent être différentes dans le temps, alors que les taux de retouches et la durée maximale de chevauchement peuvent être différents selon la nature des activités chevauchables;
- Les contraintes de ressource dans la pratique peuvent être plus flexibles. Il est possible de faire des heures supplémentaires, affecter des ressources supplémentaires ponctuellement, effectuer des quarts de travail supplémentaire. L'utilisation de tels moyens à cependant une influence sur le coût des activités.
- Nous avons considéré uniquement des relations de précédence de type fin-début pour les activités qui ne peuvent se chevaucher;
- Les méthodes de résolution pour le problème d'ordonnancement de projet avec chevauchement d'activités sont testées sur des projets jusque 120 activités, alors qu'une partie importante des projets industriels comportent plus d'activités.
- L'analyse statistique, le modèle prédictif et les recommandations générales formulées dans le Chapitre 6 sont issus des résultats de projets jusque 120 activités, alors qu'une partie importante des projets industriels comportent plus d'activités. Il n'est pas certain que ces analyses soient transposables à des projets de plus grande taille;
- Le problème d'ordonnancement de projet ne tient pas compte du processus réel de planification de projets. En effet, le RCPSP et l'extension avec le chevauchement d'activités supposent un découpage du projet et une estimation des paramètres à un niveau détaillé que constitue le niveau des activités. Pourtant, le processus de planification s'effectue dans la pratique souvent sous la forme d'une approche hiérarchique de planification constituée de niveaux stratégiques, tactiques et opérationnels qui intègre à chaque niveau plus de détails au fur et à mesure que le projet est défini (Association for the Advancement of Cost Engineering, 2010; Cherkaoui et al., 2013; Herroelen, 2005; Knutson & Bitz, 1991).

8.5 Perspectives de recherche

Les résultats obtenus et les limitations générales et spécifiques constatées dans cette thèse permettent de proposer quelques perspectives pour les recherches futures :

- Développer une démarche d'estimation des données de chevauchement basée sur les processus réels de planification et suivi de projets industriels. Tel que discuté à la section 7.4.1, nous avons pu voir chez notre partenaire industriel que l'estimation de ces données et le recueil des informations lors du suivi de l'exécution ne sont font pas nécessairement au niveau de détail des activités, mais parfois au niveau des lots de travaux et des contrats. Considérant la taille importante des projets, étudier sur des projets réels le chevauchement à des niveaux supérieurs tel qu'entre les disciplines d'ingénierie (génie civil, mécanique, électrique,…), ou globalement entre les phases d'ingénierie, approvisionnement et construction permettrait à la fois de limiter la quantité de données à estimer pour des problèmes de grande taille, et tenir compte de la structure des informations de planification et de suivi de projet en pratique.
- Analyser les processus de demandes de changement dans les projets industriels. Nous avons pu constater chez notre partenaire industriel que les modifications apportées aux dessins d'ingénierie, à la construction et aux contrats sont très documentées et suivent des processus d'autorisations bien définies. Ces processus et les documents associés pourraient être analysés pour estimer la durée des certaines retouches reliées au chevauchement. En analysant les comptes rendus de réunion, il serait aussi possible de quantifier en partie les durées de communication et de coordination. Enfin, pour les firmes comme notre partenaire indtsuriel qui utilisent des systèmes de gestion de projets intégrés, de nombreuses données de projet passées sont archivées et potentiellement accessibles pour aider à estimer les données de chevauchement;
- Confronter les recommandations de cette thèse à des avis d'experts et à l'analyse de projets réels passés;
- Résoudre le problème d'ordonnancement de projet avec chevauchement d'activités sur des projets de très grande taille. Ceci permettrait de s'assurer que les conclusions tirées dans le Chapitre 6 sur l'analyse de projets de 30 à 120 activités sont applicables à des

projets de plus grande taille, et aussi d'évaluer la capacité de la métaheuristique que nous avons développée à résoudre ce problème avec des temps de calcul acceptables.

- Modéliser de façon plus réaliste le problème d'ordonnancement de projets avec chevauchement d'activités en ce qui concerne les relations de précédence et les contraintes de ressources. En particulier, il s'agirait d'étendre le problème d'ordonnancement de projets avec chevauchement d'activités en incluant la possibilité d'itérations d'activités, de relations de précédence généralisées (Bartusch et al., 1988; De Reyck & Herroelen, 1998) et de contraintes de ressources flexibles dans le temps;
- Étudier le problème d'ordonnancement de projets avec chevauchement d'activités dans un contexte incertain. Les travaux présentés à la section 2.4.4 sur les modèles d'ordonnancement stochastique avec chevauchement d'activités visent plus à effectuer une analyse de risque basée sur la distribution de probabilité de la durée ou du coût de projet qu'à proposer une action pour faire face aux incertitudes. À partir de la revue de littérature sur le problème standard d'ordonnancement stochastique de projet présentée à la section 2.2.2, on s'aperçoit pourtant que des approches visant à faire face aux incertitudes existent. En particulier, plusieurs de ces approches se basent sur le développement au préalable d'un calendrier de base déterministe, et utilisent des méthodes approchées dérivées du problème déterministe (Artigues & Rivreau, 2008; Guéret & Jussien, 2008; Herroelen & Leus, 2004, 2005). La formulation du problème d'ordonnancement de projet avec chevauchement d'activités dans un contexte déterministe constitue ainsi un premier pas vers le développement d'approches d'ordonnancement avec incertitudes.

BIBLIOGRAPHIE

Alcaraz, J., & Maroto, C. (2001). A Robust Genetic Algorithm for Resource Allocation in Project Scheduling. *Annals of Operations Research, 102*(1-4), 83-109. doi: 10.1023/A:1010949931021

Alvarez-Valdes, R., & Tamarit , J. M. (1993). Project scheduling polyhedron: Dimension, facets and lifting theorems. *European Journal of Operational Research, 67*(2), 204-220. doi: 10.1016/0377-2217(93)90062-r

Anumba, C. J., Kamara, J. M., & Cutting-Decelle, A.-F. (Édit.). (2007). *Concurrent engineering in construction projects*. London: Taylor & Francis.

Artigues, C., Brucker, P., Knust, S., Kone, O., Lopez, P., & Mongeau, M. (2013). A note on "event-based MILP models for resource-constrained project scheduling problems". *Computers & Operations Research, 40*(4), 1060-1063. doi: 10.1016/j.cor.2012.10.018

Artigues, C., Michelon, P., & Reusser, S. (2003). Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research, 149* 249-267. doi: 10.1016/s0377-2217(02)00758-0

Artigues, C., & Rivreau, D. (2008). Heuristics. Dans C. Artigues, S. Demassey & E. Néron (Édit.), *Resource-Constrained Project Scheduling: Models, Algorithms and Applications* (p. 87-105). London, UK: Wiley.

Association for the Advancement of Cost Engineering. (2010). *Schedule Classification system* (Rapport n^{o} 27R-03).

Baar, T., Brucker, P., & Knust, S. (1998). Tabu-search algorithms and lower bounds for the resource-constrained project scheduling problem. Dans *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization* (p. 1-18). Boston, MA, USA: Kluwer Academic Publishers.

Baccarini, D. (1996). Concept of project complexity - a review. *International Journal of Project Management, 14*(4), 201-204. doi: 10.1016/0263-7863(95)00093-3

Baptiste, P., & Le Pape, C. (2000). Constraint Propagation and Decomposition Techniques for Highly Disjunctive and Highly Cumulative Project Scheduling Problems. *Constraints, 5*(1-2), 119-139. doi: 10.1023/a:1009822502231

Baradaran, S., Fatemi Ghomi, S. M. T., Mobini, M., & Hashemin, S. S. (2010). A hybrid scatter search approach for resource-constrained project scheduling problem in PERT-type networks. *Advances in Engineering Software, 41*(7-8), 966-975. doi: 10.1016/j.advengsoft.2010.05.010

Bartusch, M., Mohring, R. H., & Radermacher, F. J. (1988). Scheduling project networks with resource constraints and time windows. *Annals of Operations Research, 16*(1), 199-240.

Baydoun, G. (2014). *Un modèle de planification tactique avec chevauchement.* (Maîtrise, École Polytechnique de Montréal, Montréal, Canada).

Baydoun, G., Haït, A., Pellerin, R., Clément, B., & Bouvignies, G. (2014). A rough-cut capacity planning model with overlapping. *submitted to OR Spektrum*.

Bellenguez-Morineau, O., & Néron, E. (2008). Multi-Mode and Multi-Skill Project Scheduling Problem. Dans C. Artigues, S. Demassey & E. Néron (Édit.), *Resource-Constrained Project Scheduling: Models, Algorithms and Applications* (p. 149-160). London, UK: Wiley.

Berthaut, F., Grèze, L., Pellerin, R., Perrier, N., & Hajji, A. (2011, 25-27 mai). *Optimal resource-constraint project scheduling with overlapping modes.* Communication présentée à 4th International Conference on Industrial Engineering and Systems Management, Metz, France (p. 299-308).

Berthaut, F., Pellerin, R., Hajji, A., & Perrier, N. (2015). A Path Relinking-based Scatter Search for the Resource-Constrained Project Scheduling Problem. *submitted to European Journal of Operational Research*.

Berthaut, F., Pellerin, R., Perrier, N., & Hajji, A. (2014). Time-cost trade-offs in resource-constraint project scheduling problems with overlapping modes. *International Journal of Project Organisation and Management, 6*(3), 215-236. doi: 10.1504/IJPOM.2014.065259

Blackburn, J. D., Hoedemaker, G., & van Wassenhove, L. N. (1996). Concurrent software engineering: prospects and pitfalls. *IEEE Transactions on Engineering Management, 43*(2), 179-188. doi: 10.1109/17.509983

Blacud, N. A., Bogus, S. M., Diekmann, J. E., & Molenaar, K. R. (2009). Sensitivity of Construction Activities under Design Uncertainty. *Journal of Construction Engineering and Management, 135*(3), 19-206. doi: 10.1061/(ASCE)0733-93642009)135:3(199)

Blazewicz, J., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics, 5*(1), 11-24. doi: 10.1016/0166-218x(83)90012-4

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

Bogus, S. M., Molenaar, K. R., & Diekmann, J. E. (2005). Concurrent engineering approach to reducing design delivery time. *Journal of Construction Engineering and Management, 131*(11), 1179-1185. doi: 10.1061/(asce)0733-9364(2005)131:11(1179)

Bogus, S. M., Molenaar, K. R., & Diekmann, J. E. (2006). Strategies for overlapping dependent design activities. *Construction Management and Economics, 24*(8), 829-837. doi: 10.1080/01446190600658529

Boucherit, J. B., Pellerin, R., Berthaut, F., & Hajji, A. (2010). *A genetic algorithm for solving material space and resource constrained project scheduling problems.* Communication présentée à 12ème International Conference on The Modern Information Technology in the Innovation Processes of the Industrial Enterprises (MITIP), Aalborg University, Aalborg, Danemark (p. 19-26).

Bouleimen, K., & Lecocq, H. (2003). A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research, 149* 268-281. doi: 10.1016/s0377-2217(02)00761-0

Bourdichon, P. (1994). *L'ingénierie simultanée : et la gestion d'informations*. Paris: Hermès.

Bozejko, W., Hejducki, Z., Uchroski, M., & Wodecki, M. (2014). Solving resource-constrained construction scheduling problems with overlaps by metaheuristic. *Journal of Civil Engineering and Management, 20*(5), 649-659. doi: 10.3846/13923730.2014.906496

Browning, T. R. (2001). Applying the design structure matrix to system decomposition and integration problems: A review and new directions. *IEEE Transactions on Engineering Management, 48*(3), 292-306. doi: 10.1109/17.946528

Browning, T. R., & Eppinger, S. D. (2002). Modeling impacts of process architecture on cost and schedule risk in product development. *IEEE Transactions on Engineering Management, 49*(4), 428-442. doi: 10.1109/tem.2002.806709

Brucker, P., Drexl, A., Mohring, R., Neumann, K., & Pesch, E. (1999). Resource-constrained project scheduling: notation, classification, models, and methods. *European Journal of Operational Research, 112*(1), 3-41. doi: 10.1016/s0377-2217(98)00204-5

Brucker, P., & Knust, S. (2000). A linear programming and constraint propagation-based lower bound for the RCPSP. *European Journal of Operational Research, 127*(2), 355-362. doi: 10.1016/S0377-2217(99)00489-0

Brucker, P., & Knust, S. (2003). Lower bounds for resource-constrained project scheduling problems. *European Journal of Operational Research, 149* 302-313. doi: 10.1016/s0377-2217(02)00762-2

Brucker, P., & Knust, S. (2012). *Complex scheduling* (2e éd.). Berlin, Allemagne: Springer.

Brucker, P., Knust, S., Schoo, A., & Thiele, O. (1998). Branch and bound algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research, 107*(2), 272-288. doi: 10.1016/s0377-2217(97)00335-4

Carlier, J., Moukrim, A., & Xu, H. (2009). *A Memetic Algorithm for the Resource Constrained Project Scheduling Problem.* Communication présentée à International Conference on Industrial Engineering and Systems Management IESM, Montréal, Canada.

Carlier, J., & Pinson, E. (1989). An algorithm for solving the job-shop problem. *Management Science, 35*(2), 164-176. doi: 10.1287/mnsc.35.2.164

Carrascosa, M. (1999). *Product development timing and cost analysis using information flow modeling.* (Ph.D., Massachusetts Institute of Technology, Cambridge, Massachusetts, USA). Accessible par ProQuest Dissertations and Theses. (0800805). Tiré de http://search.proquest.com/docview/304556605?accountid=40695

Carruthers, J. A., & Battersby, A. (1966). Advances in critical path methods. *Operational Research Quarterly, 17*(4), 359-380.

Cervantes, M., Lova, A., Tormos, P., & Barber, F. (2008). A Dynamic Population Steady-State Genetic Algorithm for the Resource-Constrained Project Scheduling Problem. Dans N. T. Nguyen, L. Borzemski, A. Grzech & M. Ali (Édit.), *New Frontiers in Applied Artificial Intelligence* (p. 611-620). Berlin, Allemagne: Springer Berlin Heidelberg.

Chen, W., Shi, Y.-j., Teng, H.-f., Lan, X.-p., & Hu, L.-c. (2010). An efficient hybrid algorithm for resource-constrained project scheduling. *Information Sciences, 180*(6), 1031-1039. doi: 10.1016/j.ins.2009.11.044

Cherkaoui, K., Pellerin, R., Baptiste, P., & Perrier, N. (2013). *Planification hiérarchique de projets EPCM*. Communication présentée à 10ème Conférence Internationale de Génie Industriel CIGI, La Rochelle, France.

Cho, S.-H., & Eppinger, S. D. (2005). A simulation-based process model for managing complex design projects. *IEEE Transactions on Engineering Management, 52*(3), 316-328. doi: 10.1109/tem.2005.850722

Christofides, N., Alvarez-Valdes, R., & Tamarit, J. M. (1987). Project scheduling with resource constraints: a branch and bound approach. *European Journal of Operational Research, 29*(3), 262-273. doi: 10.1016/0377-2217(87)90240-2

Clark, K. B., & Fujimoto, T. (1991). *Product development performance : strategy, organization, and management in the world auto industry*. Boston, Mass.: Harvard Business School Press.

De Reyck, B., & Herroelen, W. (1998). A branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations. *European Journal of Operational Research, 111*(1), 152-174. doi: 10.1016/s0377-2217(97)00305-6

Debels, D., De Reyck, B., Leus, R., & Vanhoucke, M. (2006). A hybrid scatter search/electromagnetism metaheuristic for project scheduling. *European Journal of Operational Research, 169*(2), 638-653. doi: 10.1016/j.ejor.2004.08.020

Debels, D., & Vanhoucke, M. (2007). A Decomposition-Based Genetic Algorithm for the Resource-Constrained Project-Scheduling Problem. *Operations Research, 55*(3), 457-469. doi: 10.1287/opre.1060.0358

Deblaere, F., Demeulemeester, E., Herroelen, W., & Van de Vonder, S. (2006). *Proactive Resource Allocation Heuristics for Robust Project Scheduling* (Rapport n^{o} KBI 0608). KU Leuven. Tiré de http://dx.doi.org/10.2139/ssrn.870228

Demassey, S., Artigues, C., & Michelon, P. (2005). Constraint-Propagation-Based Cutting Planes: An Application to the Resource-Constrained Project Scheduling Problem. *INFORMS Journal on Computing, 17*(1), 52-65. doi: 10.1287/ijoc.1030.0043

Demeulemeester, E., & Herroelen, W. (1992). A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science, 38*(12), 1803-1818. doi: 10.1287/mnsc.38.12.1803

Demeulemeester, E., & Herroelen, W. (1997). New benchmark results for the resource-constrained project scheduling problem. *Management Science, 43*(11), 1485-1492.

Demeulemeester, E., Herroelen, W., & Leus, R. (2008). Proactive-Reactive Project Scheduling. Dans C. Artigues, S. Demassey & E. Néron (Édit.), *Resource-Constrained Project Scheduling: Models, Algorithms and Applications* (p. 203-212). London, UK: Wiley.

Deshpande, A. S. (2009). *Best practices for the management of design in fast track industrial projects*. (Ph.D., University of Cincinnati, Ann Arbor). Accessible par ProQuest Dissertations and Theses. (3368447). Tiré de http://search.proquest.com/docview/304863258?accountid=40695

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics - Part B, 26*(1), 29-41. doi: 10.1109/3477.484436

Duan, N., Newhouse, J. P., Morris, C. N., & Manning, W. G. (1983). A comparison of alternative models for the demand for medical care. *Journal of business & economic statistics, 1*(2), 115-126.

Dzeng, R.-J. (2006). Identifying a design management package to support concurrent design in building wafer fabrication facilities. *Journal of Construction Engineering and Management, 132*(6), 606-614. doi: 10.1061/(asce)0733-9364(2006)132:6(606)

Eppinger, S. D., Whitney, D. E., Smith, R. P., & Gebala, D. A. (1994). A model-based method for organizing tasks in product development. *Research in Engineering Design, 6*(1), 1-13. doi: 10.1007/bf01588087

Fang, C., & Wang, L. (2012). An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem. *Computers & Operations Research, 39*(5), 890-901. doi: 10.1016/j.cor.2011.07.010

Fayez, M., Axelsson, P., Oloufa, A. A., & Hosni, Y. (2003). *DSM Versus CPM: Issues for Planning Design Construction Activities.* Communication présentée à Contruction Research Congress, Winds of Change: Integration and Innovation in Construction, Proceedings of the Congress, Honolulu, HI., United states (p. 181-188).

Fleszar, K., & Hindi, K. S. (2004). Solving the resource-constrained project scheduling problem by a variable neighbourhood search. *European Journal of Operational Research, 155*(2), 402-413. doi: 10.1016/s0377-2217(02)00884-6

Ford, D. N., & Sterman, J. D. (2003). The Liar's Club: Concealing Rework in Concurrent Development. *Concurrent Engineering Research and Applications, 11*(3), 211-219. doi: 10.1177/106329303038028

Gerk, J. E. V., & Qassim, R. Y. (2008). Project acceleration via activity crashing, overlapping, and substitution. *IEEE Transactions on Engineering Management, 55*(4), 590-601. doi: 10.1109/tem.2008.927786

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences, 8*(1), 156-166. doi: 10.1111/j.1540-5915.1977.tb01074.x

Glover, F., Hersh G., & McMillian, C. (1977). *Selecting subset of maximum diversity* (Rapport n^{o} MS/IS 77-9). Boulder, Colorado, USA: Business Research Division, Graduate School of Business Administration, University of Colorado.

Glover, F., Kelly, J. P., & Laguna, M. (1995). Genetic algorithms and tabu search: hybrids for optimization. *Computers & Operations Research, 22*(1), 111-134. doi: 10.1016/0305-0548(93)e0023-m

Goldratt, E. M. (1997). *Critical Chain*. Great Barrington, MA, USA: North River Press.

Goncalves, J. F., Resende, M. G. C., & Mendes, J. J. M. (2011). A biased random-key genetic algorithm with forward-backward improvement for the resource constrained project scheduling problem. *Journal of Heuristics, 17*(5), 467-486. doi: 10.1007/s10732-010-9142-2

Grèze, L. (2011). *Processus d'accélération de projets sous contraintes de ressources avec modes de chevauchement.* (Maîtrise, École Polytechnique de Montréal, Montréal, Canada). Accessible par ProQuest Dissertations and Theses. Tiré de http://search.proquest.com/docview/927944960

Grèze, L., Pellerin, R., & Leclaire, P. (2011). *Processus d'accélération de projets sous contraintes de ressources avec modes de chevauchement*. Communication présentée à Conférence Internationale de Génie Industriel CIGI2011, St-Sauveur, Canada.

Grèze, L., Pellerin, R., Leclaire, P., & Perrier, N. (2014a). CIGI2011: A heuristic method for resource-constrained project scheduling with activity overlapping. *Journal of Intelligent Manufacturing, 25*(4), 787-811. doi: 10.1007/s10845-012-0719-5

Grèze, L., Pellerin, R., Leclaire, P., & Perrier, N. (2014b). Evaluating the effectiveness of task overlapping as a risk response strategy in engineering projects. *International Journal of Project Organisation and Management, 6*(1-2), 33-47. doi: 10.1504/IJPOM.2014.059743

Guéret, C., & Jussien, N. (2008). Reactive Approaches. Dans C. Artigues, S. Demassey & E. Néron (Édit.), *Resource-Constrained Project Scheduling: Models, Algorithms and Applications* (p. 191-201). London, UK: Wiley.

Hagstrom, J. N. (1988). Computational complexity of PERT problems. *Networks, 18*(2), 139-147. doi: 10.1002/net.3230180206

Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics, 45*(7), 733-750.

Hartmann, S. (1999). *Project Scheduling Under Limited Resources: Models, Methods, and Applications*. New York, NY, USA: Springer-Verlag Berlin Heidelberg.

Hartmann, S. (2002). A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics, 49*(5), 433-448.

Hartmann, S., & Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research, 207*(1), 1-14. doi: 10.1016/j.ejor.2009.11.005

Hartmann, S., & Drexl, A. (1998). Project scheduling with multiple modes: A comparison of exact algorithms. *Networks, 32*(4), 283-297. doi: 10.1002/(SICI)1097-0037(199812)32:4<283::AID-NET5>3.0.CO;2-I

Hartmann, S., & Kolisch, R. (2000). Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research, 127*(2), 394-407. doi: 10.1016/S0377-2217(99)00485-3

Herroelen, W. (2005). Project scheduling - theory and practice. *Production and Operations Management, 14*(4), 413-432. doi: 10.1111/j.1937-5956.2005.tb00230.x

Herroelen, W., De Reyck, B., & Demeulemeester, E. (1998). Resource-constrained project scheduling: A survey of recent developments. *Computers & Operations Research, 25*(4), 279-302. doi: 10.1016/S0305-0548(97)00055-5

Herroelen, W., & Leus, R. (2004). Robust and reactive project scheduling: a review and classification of procedures. *International Journal of Production Research, 42*(8), 1599-1620. doi: 10.1080/00207543310001638055

Herroelen, W., & Leus, R. (2005). Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research, 165*(2), 289-306. doi: 10.1016/j.ejor.2004.04.002

Hindi, K. S., Yang, H., & Fleszar, K. (2002). An evolutionary algorithm for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation, 6*(5), 512-518. doi: 10.1109/tevc.2002.804914

Holland, J. H. (1975). *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence*. Ann Arbor: University of Michigan Press.

Hosmer, D. W., & Lemeshow, S. (2000). *Applied logistic regression* (2nd[e] éd.). New York, NY, USA: Wiley-Interscience.

Huang, E., & Chen, S.-J. (2006). Estimation of project completion time and factors analysis for concurrent engineering project management: A simulation approach. *Concurrent Engineering Research and Applications, 14*(4), 329-341. doi: 10.1177/1063293x06072482

Icmeli-Tukel, O., & Rom, W. O. (1997). Ensuring quality in resource constrained project scheduling. *European Journal of Operational Research, 103*(3), 483-496. doi: 10.1016/s0377-2217(96)00305-0

Imai, K.-i., Ikujiro, N., & Hirotaka, T. (1985). Managing the New Product Development Process: How Japanese Companies Learn and Unlearn. Dans R. Hayes, K. Clark & C. Lorenz (Édit.), *The Uneasy Alliance: Managing the Productivity-Technology Dilemma*. Boston, MA, USA: Harvard Business School Press.

Kerzner, H. (2009). *Project management : a systems approach to planning, scheduling, and controlling* (10[e] éd.). Hoboken, N.J., USA: John Wiley & Sons.

Khoueiry, Y., Srour, I., & Yassine, A. (2013). An optimization-based model for maximizing the benefits of fast-track construction activities. *Journal of the Operational Research Society, 64*(8), 1137-1146. doi: 10.1057/jors.2013.30

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing *Science, 220*(4598), 671-680. doi: 10.2307/1690046

Klein, R. (2000). Bidirectional planning: Improving priority rule-based heuristics for scheduling resource-constrained projects. *European Journal of Operational Research, 127*(3), 619-638. doi: 10.1016/s0377-2217(99)00347-1

Klein, R., & Scholl, A. (1999). Computing lower bounds by destructive improvement: An application to resource-constrained project scheduling. *European Journal of Operational Research, 112*(2), 322-346. doi: 10.1016/s0377-2217(97)00442-6

Knutson, J., & Bitz, I. (1991). *Project Management: How to Plan and Manage Successful Projects*. New York, NY, USA: AMACOM.

Kochetov, Y., & Stolyar, A. (2003). *Evolutionary Local Search with Variable Neighborhood for the Resource Constrained Project Scheduling Problem*. Communication présentée à 3rd International Workshop on Computer Science and Information Technologies, Ufa, Russia.

Kolisch, R. (1996). Serial and parallel resource-constrained project scheduling methods revisited: theory and computation. *European Journal of Operational Research, 90*(2), 320-333. doi: 10.1016/0377-2217(95)00357-6

Kolisch, R. (1999). Resource Allocation Capabilities of Commercial Project Management Software Packages. *Interfaces, 29*(4), 19-31.

Kolisch, R., & Hartmann, S. (1999). Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis. Dans J. Węglarz (Édit.), *Project Scheduling: Recent Models, Algorithms and Applications* (p. 147-178). New York, USA: Springer Science+Business Media.

Kolisch, R., & Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research, 174*(1), 23-37. doi: 10.1016/j.ejor.2005.01.065

Kolisch, R., & Padman, R. (2001). An integrated survey of deterministic project scheduling. *Omega, 29*(3), 249-272. doi: 10.1016/S0305-0483(00)00046-3

Kolisch, R., Schwindt, C., & Sprecher, A. (1999). Benchmark instances for project scheduling problems. Dans J. Weglarz (Édit.), *Project Scheduling* (p. 197-212). New York, NY, USA: Springer Science+Business Media.

Kolisch, R., & Sprecher, A. (1997). PSPLIB - A project scheduling problem library: OR Software - ORSEP Operations Research Software Exchange Program. *European Journal of Operational Research, 96*(1), 205-216. doi: 10.1016/S0377-2217(96)00170-1

Kolisch, R., Sprecher, A., & Drexl, A. (1992). *Characterization and generation of a general class of resource-constrained project scheduling problems - Easy and hard instances* (Rapport n^{o} 301). Kiel, Allemagne: Instituten für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel.

Kolisch, R., Sprecher, A., & Drexl, A. (1995). Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science, 41*(10), 1693-1703. Tiré de http://www.jstor.org/stable/2632747

Koné, O., Artigues, C., Lopez, P., & Mongeau, M. (2011). Event-based MILP models for resource-constrained project scheduling problems. *Computers and Operations Research, 38* 3-13. doi: 10.1016/j.cor.2009.12.011

Koulinas, G., Kotsikas, L., & Anagnostopoulos, K. (2014). A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem. *Information Sciences, 277* 680-693. doi: 10.1016/j.ins.2014.02.155

Krishnan, V., Eppinger, S. D., & Whitney, D. E. (1997). A Model-based framework to overlap product development activities. *Management Science, 43*(4), 437-451.

Leus, R., & Herroelen, W. (2004). Stability and resource allocation in project planning. *IIE Transactions, 36*(7), 667-682. doi: 10.1080/07408170490447348

Li, K. Y., & Willis, R. J. (1992). An iterative scheduling technique for resource-constrained project scheduling. *European Journal of Operational Research, 56*(3), 370-379. doi: 10.1016/0377-2217(92)90320-9

Liberatore, M. J., & Pollack-Johnson, B. (2003). Factors influencing the usage and selection of project management software. *IEEE Transactions on Engineering Management, 50*(2), 164-174. doi: 10.1109/tem.2003.810821

Liberatore, M. J., & Pollack-Johnson, B. (2006). Extending project time-cost analysis by removing precedence relationships and task streaming. *International Journal of Project Management, 24*(6), 529-535. doi: 10.1016/j.ijproman.2006.04.004

Liberatore, M. J., Pollack-Johnson, B., & Smith, C. A. (2001). Project management in construction: Software use and research directions. *Journal of Construction Engineering and Management, 127*(2), 101-107. doi: 10.1061/(asce)0733-9364(2001)127:2(101)

Lim, A., Ma, H., Rodrigues, B., Tan, S. T., & Xiao, F. (2013). New meta-heuristics for the resource-constrained project scheduling problem. *Flexible Services and Manufacturing Journal, 25*(1-2), 48-73. doi: 10.1007/s10696-011-9133-0

Lim, T.-K., Yi, C.-Y., Lee, D.-E., & Arditi, D. (2014). Concurrent Construction Scheduling Simulation Algorithm. *Computer-Aided Civil and Infrastructure Engineering, 29*(6), 449-463. doi: 10.1111/mice.12073

Lin, J., Chai, K. H., Brombacher, A. C., & Wong, Y. S. (2009). Optimal overlapping and functional interaction in product development. *European Journal of Operational Research, 196*(3), 1158-1169. doi: 10.1016/j.ejor.2008.05.030

Lin, J., Qian, Y., Cui, W., & Miao, Z. (2010). Overlapping and communication policies in product development. *European Journal of Operational Research, 201*(3), 737-750. doi: 10.1016/j.ejor.2009.03.040

Loch, C. H., & Terwiesch, C. (1998). Communication and uncertainty in concurrent engineering. *Management Science, 44*(8), 1032-1048.

Manning, W. G., & Mullahy, J. (2001). Estimating log models: to transform or not to transform? *Journal of Health Economics, 20*(4), 461-494. doi: 10.1016/S0167-6296(01)00086-8

Marti, R., Laguna, M., & Glover, F. (2006). Principles of scatter search. *European Journal of Operational Research, 169*(2), 359-372. doi: 10.1016/j.ejor.2004.08.004

McCullagh, P., & Nelder, J. A. (1989). *Generalized Linear Models* (vol. 2). London, Royaume-Uni: Chapman and Hall.

Mendes, J. J. M., Goncalves, J. F., & Resende, M. G. C. (2009). A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & Operations Research, 36*(1), 92-109. doi: 10.1016/j.cor.2007.07.001

Merkle, D., Middendorf, M., & Schmeck, H. (2002). Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation, 6*(4), 333-346. doi: 10.1109/TEVC.2002.802450

Mingozzi, A., Maniezzo, V., Ricciardelli, S., & Bianco, L. (1998). An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Management Science, 44*(5), 714-729. doi: 10.1287/mnsc.44.5.714

Mobini, M., Mobini, Z., & Rabbani, M. (2011). *An Artificial Immune Algorithm for the project scheduling problem under resource constraints.* Communication présentée à Applied Soft Computing Journal, Langford Lane, Kidlington, Oxford, OX5 1GB, United Kingdom (vol. 11, p. 1975-1982). doi: 10.1016/j.asoc.2010.06.013

Mobini, M., Rabbani, M., Amalnik, M. S., Razmi, J., & Rahimi-Vahed, A. R. (2009). Using an enhanced scatter search algorithm for a resource-constrained project scheduling problem. *Soft Computing, 13*(6), 597-610. doi: 10.1007/s00500-008-0337-5

Möhring, R. H., Schulz, A. S., Stork, F., & Uetz, M. (2003). Solving project scheduling problems by minimum cut computations. *Management Science, 49*(3), 330-350. doi: 10.1287/mnsc.49.3.330.12737

Mubarak, S. A. (2010). *Construction project scheduling and control* (2e éd.). Hoboken, N.J., USA: John Wiley & Sons.

Nan, R., He, Y.-X., & Han, B.-J. (2013). Research on complex product WBS process simulation and prediction. *Information Technology Journal, 12*(16), 3880-3884. doi: 10.3923/itj.2013.3880.3884

Neumann, K. (1999). Scheduling of projects with stochastic evolution structure. Dans J. Węglarz (Édit.), *Project Scheduling: Recent Models, Algorithms and Applications* (p. 309-332). New York, USA: Springer Science+Business Media.

Nouri, N., Krichen, S., & Ladhari, T. (2013). *A discrete artificial bee colony algorithm for resource-constrained project scheduling problem.* Communication présentée à 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO), Hammamet, Tunisie (p. 1-6). doi: 10.1109/ICMSAO.2013.6552557

Özdamar, L., & Ulusoy, G. (1995). A survey on the resource-constrained project scheduling problem. *IIE Transactions, 27*(5), 574-586. doi: 10.1080/07408179508936773

Özdamar, L., & Ulusoy, G. (1996). Note on an iterative forward/backward scheduling technique with reference to a procedure by Li and Willis. *European Journal of Operational Research, 89*(2), 400-407. doi: 10.1016/0377-2217(94)00272-x

Palpant, M., Artigues, C., & Michelon, P. (2004). LSSPER: solving the resource-constrained project scheduling problem with large neighbourhood search. *Annals of Operations Research, 131* 237-257. doi: 10.1023/b:anor.0000039521.26237.62

Paraskevopoulos, D. C., Tarantilis, C. D., & Ioannou, G. (2012). Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm. *Expert Systems with Applications, 39*(4), 3983-3994. doi: 10.1016/j.eswa.2011.09.062

Patterson, J. H., Talbot, F. B., Slowinski, R., & Weglarz, J. (1990). Computational experience with a backtracking algorithm for solving a general class of precedence and resource-constrained scheduling problems. *European Journal of Operational Research, 49*(1), 68-79. doi: 10.1016/0377-2217(90)90121-q

Paul, P., Pennell, M. L., & Lemeshow, S. (2013). Standardizing the power of the Hosmer–Lemeshow goodness of fit test in large data sets. *Statistics in Medicine, 32*(1), 67-80. doi: 10.1002/sim.5525

Pellerin, R. (1997). *Modele d'ordonnancement dynamique de projets de refection.* (Ph.D., Ecole Polytechnique de Montréal, Montreal (Canada)). Accessible par ProQuest Dissertations and Theses. (NQ33018). Tiré de http://search.proquest.com/docview/304408362?accountid=40695

Pena-Mora, F., & Li, M. (2001). Dynamic planning and control methodology for design/build fast-track construction projects. *Journal of Construction Engineering and Management, 127*(1), 1-17. doi: 10.1061/(asce)0733-9364(2001)127:1(1)

Petitdemange, C. (1991). *La maîtrise de la valeur : la gestion de projet et l'ingénierie simultanée* (2e éd.). Paris: Association française de normalisation (Afnor).

Petrovic, R. (1968). Optimization of resource allocation in project planning. *Operations Research, 16*(3), 559-568.

Picouleau, C. (1995). New complexity results on scheduling with small communication delays. *Discrete Applied Mathematics, 60*(1-3), 331-342. doi: 10.1016/0166-218X(94)00063-J

Pinson, E., Prins, C., & Rullier, F. (1994). *Using tabu search for solving the resource-constrained project scheduling problem.* Communication présentée à 4ème International Workshop on Project Management and Scheduling, Leuven, Belgique (p. 102-106).

Portioli-Staudacher, A., Van Landeghem, H., Mappelli, M., & Redaelli, C. E. (2003). Implementation of concurrent engineering: a survey in Italy and Belgium. *Robotics and Computer-Integrated Manufacturing, 19*(3), 225-238. doi: 10.1016/s0736-5845(03)00029-2

Pritsker, A. A. B., Waiters, L. J., & Wolfe, P. M. (1969). Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach. *Management Science, 16*(1), 93-108.

Proon, S., & Jin, M. (2011). A genetic algorithm with neighborhood search for the resource-constrained project scheduling problem. *Naval Research Logistics, 58*(2), 73-82. doi: 10.1002/nav.20439

Ranjbar, M. (2008). Solving the resource-constrained project scheduling problem using filter-and-fan approach. *Applied Mathematics and Computation, 201*(1-2), 313-318. doi: 10.1016/j.amc.2007.12.025

Ranjbar, M., De Reyck, B., & Kianfar, F. (2009). A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling. *European Journal of Operational Research, 193*(1), 35-48. doi: 10.1016/j.ejor.2007.10.042

Ranjbar, M., & Kianfar, F. (2009). A hybrid scatter search for the RCPSP. *Scientia Iranica, 16*(1), 11-18.

Remington, K., & Pollack, J. (2007). *Tools for complex projects*. Aldershot, England: Gower.

Roemer, T. A., & Ahmadi, R. (2004). Concurrent crashing and overlapping in product development. *Operations Research, 52*(4), 606-622. doi: 10.1287/opre.1040.0125

Roemer, T. A., Ahmadi, R., & Wang, R. H. (2000). Time-cost trade-offs in overlapped product development. *Operations Research, 48*(6), 858-865. doi: 10.1287/opre.48.6.858.12396

Rosenkrantz, D. J., Tayi, G. K., & Ravi, S. S. (2000). Facility Dispersion Problems under Capacity and Cost Constraints. *Journal of Combinatorial Optimization, 4*(1), 7-33. doi: 10.1023/a:1009802105661

Sabbagh, K. (1996). *21st century jet : the making and marketing of the Boeing 777*. New York ; Toronto: Scribner.

Sadeghi, A., Kalanaki, A., Noktehdan, A., Samghabadi, A. S., & Barzinpour, F. (2011). Using bees algorithm to solve the resource constrained project scheduling problem in PSPLIB. Dans Q. Zhou (Édit.), *Theoretical and Mathematical Foundations of Computer Science, Communications in Computer and Information Science* (p. 486-494). Berlin, Allemagne: Springer-Verlag Berlin Heidelberg

Sprecher, A. (2000). Scheduling resource-constrained projects competitively at modest memory requirements. *Management Science, 46*(5), 710-723.

Sprecher, A. (2002). Network decomposition techniques for resource-constrained project scheduling. *Journal of the Operational Research Society, 53*(4), 405-414. doi: 10.1057/palgrave/jors/2601308

Sprecher, A., & Drexl, A. (1998). Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm1. *European Journal of Operational Research, 107*(2), 431-450. doi: 10.1016/S0377-2217(97)00348-2

Sprecher, A., Hartmann, S., & Drexl, A. (1997). An exact algorithm for project scheduling with multiple modes. *Operations-Research-Spektrum, 19*(3), 195-203.

Steward, D. V. (1981). The design structure system: a method for managing the design of complex systems. *IEEE Transactions on Engineering Management, 28*(3), 71-74.

Stinson, J. P., Davis, E. W., & Khumawala, B. M. (1978). Multiple resource-constrained scheduling using branch and bound. *AIIE Transactions, 10*(3), 252-259. doi: 10.1080/05695557808975212

Talbi, E.-G. (2009). *Metaheuristics : from design to implementation*. Hoboken, N.J.: John Wiley & Sons.

Tavares, L. V. (2002). A review of the contribution of Operational Research to Project Management. *European Journal of Operational Research, 136*(1), 1-18. doi: 10.1016/s0377-2217(01)00097-2

Terwiesch, C., & Loch, C. H. (1999). Measuring the effectiveness of overlapping development activities. *Management Science, 45*(4), 455-465.

Thammano, A., & Phu-ang, A. (2012). A hybrid evolutionary algorithm for the resource-constrained project scheduling problem. *Artificial Life and Robotics, 17*(2), 312-316. doi: 10.1007/s10015-012-0065-x

Tormos, P., & Lova, A. (2001). A competitive heuristic solution technique for resource-constrained project scheduling. *Annals of Operations Research, 102* 65-81. doi: 10.1023/a:1010997814183

Tyagi, S. K., Yang, K., & Verma, A. (2013). Non-discrete ant colony optimisation (NdACO) to optimise the development cycle time and cost in overlapped product development. *International Journal of Production Research, 51*(2), 346-361. doi: 10.1080/00207543.2011.633120

Valls, V., Ballestin, F., & Quintanilla. (2004). A Population-Based Approach to the Resource-Constrained Project Scheduling Problem. *Annals of Operations Research, 131*(1-4), 305-324. doi: 10.1023/B:ANOR.0000039524.09792.c9

Valls, V., Ballestin, F., & Quintanilla, S. (2005). Justification and RCPSP: A technique that pays. *European Journal of Operational Research, 165* 375-386. doi: 10.1016/j.ejor.2004.04.008

Valls, V., Ballestin, F., & Quintanilla, S. (2008). A hybrid genetic algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research, 185*(2), 495-508. doi: 10.1016/j.ejor.2006.12.033

Valls, V., Quintanilla, S., & Ballestin, F. (2003). Resource-constrained project scheduling: A critical activity reordering heuristic. *European Journal of Operational Research, 149* 282-301. doi: 10.1016/s0377-2217(02)00768-3

Van de Vonder, S., Demeulemeester, E., & Herroelen, W. (2004). *An investigation of efficient and effective predictive-reactive project scheduling procedures* (Rapport n^{o} 0466). KU Leuven.

Van de Vonder, S., Demeulemeester, E., & Herroelen, W. (2005). *Heuristic procedures for generating stable project baseline schedules* (Rapport n^{o} 0516). KU Leuven. Tiré de http://dx.doi.org/10.2139/ssrn.875289

Wang, H., Li, T., & Lin, D. (2010). Efficient genetic algorithm for resource-constrained project scheduling problem. *Transactions of Tianjin University, 16*(5), 376-382. doi: 10.1007/s12209-010-1495-y

Wang, J., & Lin, Y.-I. (2009). An overlapping process model to assess schedule risk for new product development. *Computers and Industrial Engineering, 57*(2), 460-474. doi: 10.1016/j.cie.2007.12.013

Williams, G. V. (1995). Fast Track Pros and Cons: Considerations for Industrial Projects. *Journal of Management in Engineering, 11*(5), 24-32.

Williams, T. M. (1999). Need for new paradigms for complex projects. *International Journal of Project Management, 17*(5), 269-273. doi: 10.1016/s0263-7863(98)00047-7

Williams, T. M. (2002). *Modelling complex projects*. Chichester, Angleterre: John Wiley & Sons.

Winner, R. I., Pennell, J. P., Bertrand, H. E., & Slusarczuk, M. M. (1988). *The role of concurrent engineering in weapons system acquisition*. DTIC Document.

Wooldridge, J. M. (2002). *Econometric Analysis of Cross Section and Panel Data*. Cambridge, MA, USA: MIT Press.

Yang, Y., & Simpson, D. (2010). Unified computational methods for regression analysis of zero-inflated and bound-inflated data. *Computational Statistics and Data Analysis, 54*(6), 1525-1534. doi: 10.1016/j.csda.2009.12.012

Zamani, R. (2010a). An accelerating two-layer anchor search with application to the resource-constrained project scheduling problem. *IEEE Transactions on Evolutionary Computation, 14*(6), 975-984. doi: 10.1109/tevc.2010.2047861

Zamani, R. (2010b). A parallel complete anytime procedure for project scheduling under multiple resource constraints. *International Journal of Advanced Manufacturing Technology, 50*(1-4), 353-362. doi: 10.1007/s00170-009-2483-z

Zamani, R. (2012). A polarized adaptive schedule generation scheme for the resource-constrained project scheduling problem. *RAIRO - Operations Research, 46*(1), 23-39. doi: 10.1051/ro/2012006

Zamani, R. (2013). A competitive magnet-based genetic algorithm for solving the resource-constrained project scheduling problem. *European Journal of Operational Research, 229*(2), 552-559. doi: 10.1016/j.ejor.2013.03.005