Nouvelles approches pour la résolution du problème d'ordonnancement de projet à moyens limités

Oumar Koné

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *l'Université Toulouse III - Paul Sabatier*
Discipline ou spécialité : *Systèmes Industriels*

Présentée et soutenue par *Oumar KONE*
Le *07 décembre 2009*

Titre : *Nouvelles approches pour la résolution du problème d'ordonnancement de projet à moyens limités*

JURY
Pierre LOPEZ et Marcel MONGEAU (Directeurs de thèse)
Jean-Charles BILLAUT (Rapporteur)
Jacques CARLIER (Rapporteur)
Christian ARTIGUES (Examinateur)
Jean-Baptiste HIRIART-URRUTY (Examinateur)
Benard PENZ (Examinateur)
Marie-Claude PORTMANN (Examinateur)

Ecole doctorale : *Ecole Doctorale Systèmes*
Unité de recherche : *LAAS-CNRS (groupe MOGISA)*
Directeur(s) de Thèse : *Pierre LOPEZ / Marcel MONGEAU*
Rapporteurs : *Jean-Charles BILLAUT / Jacques CARLIER*

TABLE DES MATIÈRES

TABLE DES MATIÈRES . . . . . **iii**

LISTE DES TABLEAUX . . . . . **vii**

LISTE DES FIGURES . . . . . **ix**

INTRODUCTION **2**

CHAPITRE 1 : LE RCPSP **6**

1.1 Définition du RCPSP 8

1.2 Description du problème 8

1.3 Complexité 11

1.4 Exemple de problème 12

1.5 Instances 14

1.6 Applications industrielles 14

1.7 Méthodes de résolution des problèmes d'OC et d'ordonnancement . . . 15

1.8 Méthodes de résolution pour le RCPSP 18

1.8.1 Le calcul de bornes inférieures 18

1.8.2 Heuristiques et métaheuristiques 21

1.8.3 Les méthodes de résolution exactes 25

1.8.4 Les modèles utilisant la programmation par contraintes (PPC) . 27

1.8.5 Les modèles utilisant la PLNE 28

1.9 Quelques extensions du RCPSP 28

1.10 Conclusion 29

CHAPITRE 2 : UTILISATION DE LA PLNE POUR LE RCPSP **32**

2.1 La programmation linéaire en nombres entiers 32

2.2 Résolution du RCPSP par la PLNE 34

2.3 Les formulations à temps discret 35

2.3.1 Formulation basique à temps discret (DT) 35
2.3.2 Formulation désagrégée à temps discret (DDT) 38
2.3.3 Les formulations à temps continu 40
2.3.4 Formulation à temps continu basée sur les flots (FCT) 41
2.3.5 Formulation 42
2.3.6 Description des contraintes 43
2.3.7 Caractéristiques du modèle 44
2.4 Conclusion 44

CHAPITRE 3 : FORMULATIONS PLNE BASÉES SUR LES ÉVÉNEMENTS 46
3.1 Notion d'événement 47
3.2 Formulation *Start/End* basée sur les événements (SEE) 48
3.2.1 Formulation 50
3.2.2 Description des contraintes 50
3.2.3 Caractéristiques du modèle 52
3.3 Formulation *On/Off* basée sur les événements (OOE) 53
3.3.1 Formulation 55
3.3.2 Description des contraintes 55
3.3.3 Caractéristiques du modèle 57
3.4 Etude analytique comparative des formulations basées sur les événements 58
3.5 Variantes de formulations et "preprocessing" 60
3.5.1 Formulations à dates d'événements distinctes 60
3.5.2 "Preprocessing" pour la formulation OOE 61
3.6 Conclusion 64

CHAPITRE 4 : LE RCPSP AVEC PRODUCTION RESSOURCES 66
4.1 Description du problème et intérêt pratique 67
4.2 Formulations à temps discret DT et DDT avec production de ressources 70
4.3 Formulation basée sur les flots FCT avec production de ressources . . . 71
4.4 Formulations SEE avec production et consommation de ressources . . . 72
4.5 Formulations OOE avec production et consommation de ressources . . 73

4.6 Conclusion 78

CHAPITRE 5 : EXPÉRIMENTATIONS 80

5.1 Le prétraitement des fenêtres de temps 80

5.2 Les indicateurs d'instances et les instances 82

5.2.1 Les indicateurs d'instances 82

5.2.2 Les instances 85

5.2.3 Les instances modifiées 86

5.2.4 Caractéristiques des instances 88

5.3 Résultats numériques 90

5.3.1 RCPSP sans production/consommation de ressources 90

5.3.2 RCPSP avec production/consommation de ressources 97

5.4 Conclusion 101

CHAPITRE 6 : LE PROBLÈME DE TRANSBORDEMENT 104

6.1 Définitions et généralités 105

6.1.1 Définition 105

6.1.2 Applications industrielles 107

6.1.3 État de l'art 107

6.2 Description du problème traité 110

6.3 Résolution du problème par branch-and-bound 112

6.4 Résultats 116

6.5 Conclusion 117

CONCLUSION GÉNÉRALE & PERSPECTIVES 120

BIBLIOGRAPHIE 122

LISTE DES TABLEAUX

1.I Exemple d'instance de RCPSP 12

3.I Instance illustrative 58
3.II Les variables x_{it} des modèles DT et DDT 59
3.III Les variables x_{ie} et y_{ie} du modèle SEE 59
3.IV Les variables z_{ie} du modèle OOE 59

4.I Comportement des variables 76
4.II Interaction des contraintes 78

5.I Valeurs moyennes des indicateurs pour chaque type d'instance utilisée . 88
5.II Résultats de la relaxation continue 91
5.III Résultats de la résolution exacte (branch and bound) 94
5.IV Synthèse des résultats des expériences 96
5.V Résultats de la résolution exacte des variantes de OOE 98
5.VI Résultats de la résolution exacte du RCPSP avec production de ressource 100

6.I Temps moyen de résolution (en seconde) 116

LISTE DES FIGURES

1.1 Graphe de précédence 13
1.2 Diagramme de Gantt 13

3.1 Exemple d'ordonnancement à temps discret 48
3.2 Exemple de diagramme de Gantt utilisant des événements 49
3.3 Exemple de diagramme de Gantt de la formulation *On/Off* 53
3.4 Diagramme de Gantt avec le temps discrétisé et les événements 59

4.1 Différentes formes de production de ressources 69
4.2 Domaine des valeurs de la variable P_{iep} 77

6.1 Exemple de "crossdock" (entrepôt de transbordement) 106
6.2 Réseau des états et espace de solutions réalisables 114
6.3 Frontière de Pareto 115
6.4 Courbes de comparaison des temps de résolution 117

INTRODUCTION

Un problème d'ordonnancement consiste à organiser l'exécution d'un ensemble d'activités soumises à des contraintes de temps et de ressources. Au sein de ce type de problèmes, un des plus généraux est le problème d'ordonnancement de projet à moyens limités ou RCPSP ("Resource-Constrained Project Scheduling Problem").

Le RCPSP est un problème NP-difficile au sens fort qui recouvre dans sa généricité un grand nombre de problèmes théoriques d'ordonnancement comme les problèmes d'atelier (Job-Shop, Flow-Shop, Flow-Shop Hybride…), les problèmes à machines parallèles ou les problèmes d'ordonnancement cumulatifs (CuSP). Il se rencontre dans un grand nombre d'applications industrielles (par exemple la découpe de tissu), informatiques (gestion des processus sur des machines parallèles ou multiprocesseurs), et organisationnelles pour ne citer que ceux-là. Il suscite donc un intérêt réel dans la communauté des chercheurs opérationnels, qui lui a consacré une littérature importante.

Dans ce problème, il s'agit ordonnancer des activités, sur des ressources renouvelables disponibles en quantité limitée. Les activités sont liées entre elles par des relations de précédence, qui expriment qu'une activité i ne peut pas commencer tant que l'activité j qui la précède n'est pas achevée. L'objectif est de déterminer une solution qui minimise la date de fin du projet, en respectant à la fois les contraintes de précédence et les contraintes de ressources, qui stipulent qu'à tout instant du projet, on ne peut utiliser plus de quantité de chaque ressource que sa quantité disponible (la somme des besoins des activités en cours d'exécution ne doit pas dépasser la capacité de la ressource).

La résolution de ce problème d'optimisation combinatoire difficile a déjà fait l'objet de nombreuses études. Les travaux visant à résoudre ce problème portent sur les méthodes de résolution exacte, le calcul de bornes inférieures et les méthodes approchées. Ces travaux s'appuient sur des outils théoriques divers comme la programmation linéaire en nombres entiers (PLNE), la programmation par contraintes, les heuristiques et métaheuristiques, et les méthodes exactes de type Branch-and-Bound.

En dépit de tous les travaux effectués pour la résolution de ce problème, il n'existe

pas de méthodes exactes permettant de résoudre systématiquement des instances de plus de soixante activités. La principale contribution scientifique de notre travail de thèse consiste à proposer de nouvelles formulations mathématiques du problème, basées essentiellement sur la PLNE.

Dans le cadre de cette thèse, nous nous intéressons donc principalement à la modélisation et la résolution du RCPSP. Pas uniquement toutefois, puisque dans le dernier chapitre de ce manuscrit, nous abordons un autre type de problème qui est celui du transbordement (“crossdocking”) dans le contexte large de la gestion de la chaîne logistique. Nous restons tout de même dans la problématique de l’ordonnancement puisque, dans ce problème de *crossdocking*, nous nous concentrons sur l’aspect ordonnancement des opérations de manutention au sein des entrepôts.

Ainsi, ce rapport sera structuré de la manière suivante. Dans le chapitre I, nous présentons en détail le problème d’ordonnancement de projet à ressources limitées, puis quelques-unes de ses applications dans le domaine industriel. Ensuite, nous présentons les techniques de résolution qui lui ont été appliquées et, dans une revue de l’état de l’art, nous énumérons quelques travaux de résolution et extensions de ce problème.

Dans le chapitre II, après une brève introduction sur la PLNE, nous abordons les formulations de type PLNE connues dans la littérature. Trois d’entre elles, dont deux basées sur une discrétisation de l’horizon et la troisième sur un horizon à temps continu, attirant notre attention feront l’objet d’une étude plus fine.

Le chapitre III est consacré à la proposition de nouvelles formulations. Ainsi, à l’aide de variables binaires indicées sur le concept d’*événements*, nous proposons deux formulations inédites à temps continu, différentes dans leur manière de marquer la période d’exécution de chaque activité.

Dans le chapitre IV, nous traitons de l’extension du RCPCP, dite *RCPSP avec pro-*

duction de ressources, qui vise à prendre en compte des ressources non-renouvelables consommées et produites lors du début et de la fin de l'exécution des activités. Nous proposons une adaptation des trois modèles présentés dans le chapitre II et des deux nouveaux modèles proposés dans le chapitre III.

Le chapitre V est dédié aux expérimentations numériques. Nous y faisons une analyse des caractéristiques d'instances à travers certains indicateurs. Après une description des conditions de tests, nous présentons, puis commentons les résultats obtenus.

Le dernier chapitre traite du problème de transbordement, également appelé crossdocking. Il s'agit d'un problème assez différent du RCPSP mais la spécificité du problème auquel nous nous attaquons dans cette partie distincte porte sur l'ordonnancement des opérations de manutention à l'intérieur d'un entrepôt de crossdocking à une seule porte d'entrée et une seule porte de sortie. Un branch-and-bound est proposé pour le résoudre, et évalué.

Nous terminons ce manuscrit de thèse par une conclusion générale sur le sujet abordé et les avancées obtenues, ainsi que les perspectives envisagées sur une suite à donner à ce travail de recherche.

CHAPITRE 1

LES PROBLÈMES D'ORDONNANCEMENT DE PROJET SOUS CONTRAINTES DE RESSOURCES

La *gestion de projets*, ensemble de techniques et méthodes visant à structurer, assurer et optimiser la bonne marche d'un projet, est une pratique qui remonte à des périodes très anciennes (depuis le temps de l'empire maya) où des projets exceptionnellement complexes étaient gérés, en utilisant principalement le bon sens et des outils très simples.

On définit un *projet* par un ensemble d'activités (ou tâches) ayant des caractéristiques propres, qui sont exécutées grâce à un ensemble de ressources.

De manière générale, une *activité* est caractérisée par sa *durée*, sa *consommation* en ressources, son *coût* et sa *priorité*.

Avec d'une part l'avènement des ordinateurs et le développement des techniques de résolution de problèmes, et, d'autre part, les contraintes économiques actuelles, la *gestion de projet* acquiert davantage d'importance dans la société présente. Dans sa pratique, elle comprend les différentes étapes pour parvenir à la réalisation d'un projet, et même au-delà, avec les processus de vérification et de contrôle de sa réalisation. Elle couvre dans l'ordre chronologique les phases de planification, d'ordonnancement et de contrôle.

Explicitement, dans sa phase de *planification*, les ressources et les activités sont identifiées et des évaluations de leurs caractéristiques sont entreprises. La phase d'*ordonnancement* proprement dite (qui nous intéresse ici) consiste à programmer l'exécution dans le temps des activités, en leur attribuant les ressources nécessaires, sans violer les contraintes imposées (les contraintes de ressources et de précédences par exemple). Enfin, le *contrôle* étudie les différences qui peuvent apparaître entre l'ordonnancement prévu et la performance courante du projet ainsi planifié.

Le *problème d'ordonnancement* consiste à organiser dans le temps la réalisation des activités d'un projet, compte tenu de contraintes temporelles (délais, contraintes d'enchaînement . . .) et de contraintes portant sur l'utilisation et la disponibilité de ressources requises pour la réalisation de ce projet. Ainsi, *un ordonnancement* constitue une solution au problème d'ordonnancement. Il décrit l'exécution des activités et l'allocation des ressources au cours du temps, satisfaisant les contraintes et s'approchant au mieux des objectifs à atteindre. De manière plus précise, on parle d'ordonnancement lorsqu'on parvient à fixer les dates de début ou de fin de chacune des activités du projet. En outre, on réserve le terme de *séquencement* au cas où seul l'ordre relatif des activités (séquence) est fixé, indépendamment de leurs dates d'exécution.

Globalement, on distingue deux grands groupes de problèmes d'ordonnancement en fonction du mode d'utilisation des ressources mises en jeu dans la réalisation du projet : *les problèmes d'ordonnancement cumulatif* et *les problèmes d'ordonnancement disjonctif*.

- Dans les *problèmes d'ordonnancement cumulatif (CuSP),* les ressources peuvent exécuter plusieurs activités simultanément. Ce faisant, dans le cadre de ressources disponibles en quantité limitée, celles-ci sont utilisées dans la limite de la quantité disponible.
- Dans les *problèmes d'ordonnancement disjonctif,* une ressource exécute exclusivement une et une seule activité à la fois. C'est le cas des problèmes d'ordonnancement à une machine (mono-machine).

Dans ce chapitre, nous nous intéressons au problème d'ordonnancement de projet à moyens limités (RCPSP : Resource-Constrained Project Scheduling Problem), qui est l'un des problèmes d'ordonnancement cumulatif les plus connus, du fait de l'intérêt que lui ont accordé les chercheurs du domaine de la recherche opérationnelle et de ses nombreuses applications industrielles.

Après une description détaillée du problème et des notations que nous utiliserons tout au long de ce document, nous présentons sa formulation conceptuelle, la complexité du problème et quelques-unes de ses applications industrielles. Par la suite, nous exhibons quelques techniques et méthodes utilisées pour sa résolution et pour finir, nous abordons quelques extensions de ce type de problème.

1.1 Définition du RCPSP

Formellement, le RCPSP est un problème particulier d'optimisation combinatoire, c'est-à-dire qu'il est défini par un espace de solution $\mathscr{X}$, qui est soit discret ou soit réductible à un ensemble discret, et par un sous-ensemble de solutions réalisables $\mathscr{Y} \subseteq \mathscr{X}$ associé à une fonction-objectif $f : \mathscr{Y} \rightarrow \mathbb{R}$. Un problème d'optimisation combinatoire vise à trouver une solution réalisable $y \in \mathscr{Y}$ telle que $f(y)$ est optimal (minimal ou maximal). Ainsi, un problème d'ordonnancement de projet sous contraintes de ressources (RCPSP) est un problème d'optimisation combinatoire défini par un un 6-uplet (V, p, E, R, B, b), où V est un ensemble d'*activités*, p est un vecteur de *durées d'exécution*, E est un ensemble de *relations de précédences*, R est un ensemble de *ressources*, B est un vecteur *capacités de ressource (disponibilités des ressources)*, et b est une matrice de *demandes (consommations de ressource)*.

1.2 Description du problème

Soit n le nombre d'activités à ordonnancer, et m le nombre de ressources disponibles. Les activités constituant le projet sont identifiées par un ensemble $\{0, \ldots, n+1\}$. L'activité 0 représente par convention le début de l'ordonnancement (début du projet), et l'activité $n+1$ quant à elle représente symétriquement la fin de l'ordonnancement (fin du projet). Toutes deux sont des activités fictives appelées également *activités jalons*. L'ensemble des activités non-fictives est identifié par $A = \{1, \ldots, n\}$. Les durées d'exécution sont représentées par un vecteur p de $\mathbb{N}^{n+2}$, où la $i^{ème}$ composante, p_i, est la durée d'exécution de l'activité i, avec les valeurs spéciales $p_0 = p_{n+1} = 0$, caractéristiques des deux activités fictives.

Les *relations de précédences* sont données par un ensemble E de paires d'indices d'activités telles que $(i, j) \in E$ signifie que l'activité i précède l'activité j. On considère connu le *graphe de précédences potentiels-tâches* $G(V,E)$ où les nœuds correspondent aux activités $V = A \cup \{0, n+1\}$, et les arcs correspondent aux relations de précédences. Ainsi, on peut identifier dans le graphe, chaque activité avec le nœud correspondant. Nous supposons que G ne contient pas de cycle, sinon les relations de précédences deviennent évidemment incohérentes. Étant donné que la précédence est une relation binaire transitive, l'existence d'un chemin dans G du nœud i au nœud j signifie aussi que l'activité i doit précéder l'activité j. Par conséquent, tous les graphes de précédence ayant la même fermeture transitive définissent les mêmes contraintes de précédences. Tenant compte de la remarque précédente, nous supposons que E est tel que l'activité 0 est un prédécesseur de toutes les autres activités, et $n+1$ est un successeur de toutes les autres activités.

On parlera de *ressource cumulative* si la ressource peut être utilisée au même moment par plusieurs activités lors de leur exécution. Dans le cas contraire, on parlera de *ressource disjonctive*. Ces ressources seront dites *renouvelables* si elles redeviennent disponibles dans leur quantité initiale, une fois libérées par les activités. Le RCPSP est qualifié de problème d'ordonnancement cumulatif car il met en jeu des ressources cumulatives et renouvelables (dans sa version basique) et ces ressources renouvelables sont formalisées par l'ensemble $R = \{1, \ldots, m\}$.

Les *disponibilités* des ressources sont représentées par un vecteur B de $\mathbb{N}^m$ tel que B_k indique la disponibilité (capacité) de la ressource k. En particulier, une ressource k telle que $B_k = 1$ est appelée *ressource unaire* (disjonctive).

Les *consommations des activités* pour les ressources figurent dans la matrice b, de taille $(n+2) \times m$, telle que le terme b_{ik} représente la quantité de ressource k utilisée par l'activité i sur toute sa période d'exécution. Notons également que $b_{0k} = b_{n+1,k} = 0$, pour tout $k \in R$.

Un *ordonnancement* est un point S de $\mathbb{R}^{n+2}$ tel que sa $i^{ème}$ composante, S_i, représente la date de début de l'activité i. S_0 est un point de référence pour le démarrage du projet. Ici, nous supposons que $S_0 = 0$. Une solution S est dite *réalisable* si elle est compatible avec les contraintes de précédences

$$S_j - S_i \geq p_i \quad \forall (i,j) \in E, \tag{1.1}$$

et les contraintes de ressources

$$\sum_{i \in A_t} b_{ik} \leq B_k \quad \forall k \in R, \forall t \in H, \tag{1.2}$$

où $A_t = \{i \in A \,|S_i \leq t < S_i + p_i\}$ représente l'ensemble des activités non-fictives en cours d'exécution à l'instant t. $H = \{0, 1, \ldots, T\}$ est *l'horizon d'ordonnancement,* et T (la longueur de l'horizon d'ordonnancement) pouvant être considérée comme une borne supérieure pour la durée totale du projet, ou *makespan*. Le makespan d'un ordonnancement S est égal à S_{n+1}, c'est-à-dire la date de début de l'activité de fin de projet. L'ensemble A_t défini ci-dessus et les contraintes impliquent qu'une activité ne peut être interrompue une fois qu'elle est commencée. Il s'agit donc de ne pas autoriser la *préemption* des activités. De ce fait, on peut alors définir le RCPSP comme suit : Le *RCPSP* est le problème visant à trouver un ordonnancement non-préemptif S de makespan minimal S_{n+1} soumis à des contraintes de précédences et des contraintes de ressources.

Un ordonnancement sera dit *semi-actif* si aucune activité ne peut être avancée sans changer l'ordre d'exécution d'une autre activité et sans violer des contraintes.

Un *ordonnancement actif* est un ordonnancement semi-actif où l'on ne peut avancer l'exécution d'une activité sans en retarder une autre.

La *formulation conceptuelle* du RCPSP se présente donc comme suit :

$$\min S_{n+1} \tag{1.3}$$

$$S_j - S_i \geq p_i \quad \forall (i,j) \in E, \tag{1.1}$$

$$\sum_{i \in A_t} b_{ik} \leq B_k \quad \forall k \in R, \forall t \in H, \tag{1.2}$$

$$S_i \geq 0.$$

En outre, on peut associer à chaque activité i une date de début au plus tôt ES_i et une date de début au plus tard LS_i, celles-ci pouvant être calculées de façon indépendante en prétraitement. ES_i est une date avant laquelle l'activité i ne peut débuter, alors que LS_i est une date après laquelle elle ne peut commencer. Ainsi, on peut établir le relation suivante :

$$ES_i \leq S_i \leq LS_i \quad \forall i \in A. \tag{1.4}$$

Cette contrainte, bien que redondante, peut être introduite sous forme de coupe pour améliorer les performances des différents modèles de programmation linéaire en nombres entiers proposés pour résoudre le RCPSP.

1.3 Complexité

En accord avec la théorie complexité [41], le RCPSP est l'un des problèmes d'optimisation combinatoire les plus difficiles à résoudre. En effet, il appartient à la classe des problèmes *NP-difficiles au sens fort*. La théorie de la complexité stipule qu'un problème d'optimisation est NP-difficile au sens fort, si sa version de problème de décision qui peut lui être associée est *NP-complet* au sens fort. La démonstration de sa NP-difficulté au sens fort est donnée par Garey et Johnson [41] et également par Blazewicz et al. [16]. Ainsi, Garey et Johnson ont démontré que le seul problème d'ordonnancement à contraintes de ressources, sans contrainte de précédence et à une seule ressource, par réduction au problème de 3-partition est déjà un problème NP-difficile au sens fort.
En revanche, sans contraintes de ressources, le problème devient facile et peut être résolu

en temps polynomial.

1.4 Exemple de problème

Le tableau 1.I ci-dessous correspond à un exemple d'instance de RCPSP. Cette instance comporte 12 activités et 2 ressources. Les activités 0 et 11 sont des activités fictives de durée opératoire nulle.

Activité	p_i	$b_{i,1}$	$b_{i,2}$	$Succ_i$
0	0	0	0	2
1	7	0	2	3
2	3	2	1	6,7
3	5	3	3	4,9
4	5	3	2	11
5	6	2	1	1
6	4	1	0	1
7	5	1	3	5,8
8	4	1	1	10
9	3	1	1	4
10	7	3	1	9
11	0	0	0	0

Tableau 1.I – Exemple d'instance de RCPSP

On rappelle les notations :
p_i : durée opératoire de l'activité i ;
$b_{i,1}$ (respectivement $b_{i,2}$) : consommation de la ressource 1 (respectivement. ressource 2) par l'activité i ;
$Succ_i$: activités successeurs de l'activité i.

La figure 1.1 représente le graphe de précédence associé au problème présenté ci-dessus. Dans ce graphe potentiels-tâches, les activités sont représentées par des cercles appelés sommets du graphe. Les valeurs $\{b_{i,1}, b_{i,2}\}$ associées aux sommets représentent la consommation de l'activité i pour la ressource 1 et pour la ressource 2, respectivement Les relations de précédence sont matérialisées par des arcs. La valeur associée à chaque

arc représente la durée opératoire de l'activité associée au sommet se trouvant à l'extrémité initiale de l'arc.

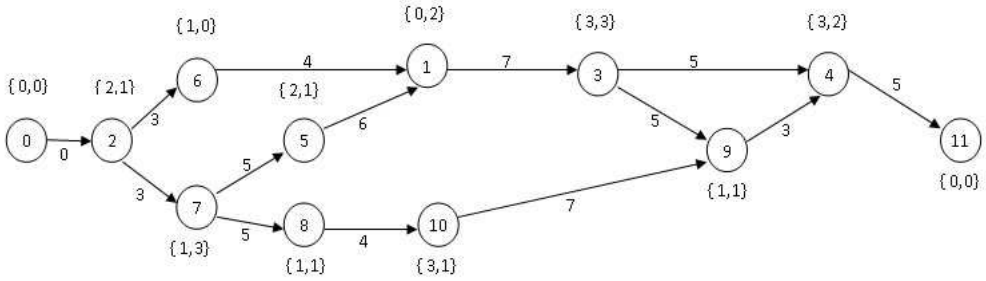

Figure 1.1 – Graphe de précédence

Par exemple, l'activité 2 qui a une durée opératoire de 3 unités de temps, est précédée de l'activité 0 et précède à son tour les activités 6 et 7.

La figure 1.2 représente, sous forme de diagramme de Gantt, un ordonnancement réalisable du problème précédent. Les ressources consommées sont en ordonnée et le temps en abscisse. Cet ordonnancement nous donne une solution avec un makespan, noté C_{max}, égal à 38.

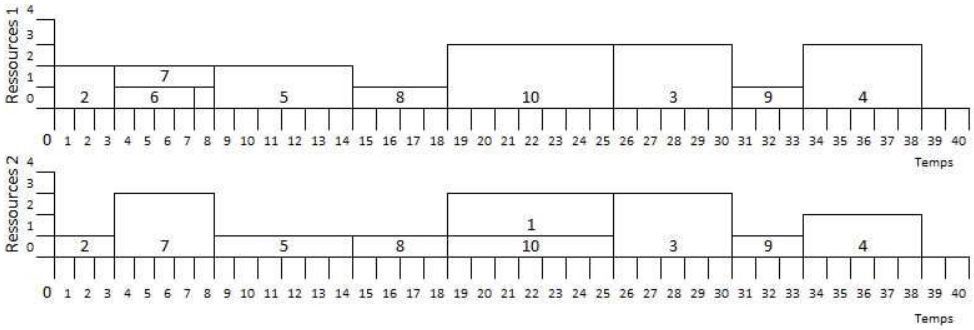

Figure 1.2 – Diagramme de Gantt

1.5 Instances

Des jeux de données (*instances*) sont disponibles pour éprouver les performances des modèles proposés pour la résolution du RCPSP (“benchmarking”). La plupart de ces instances sont disponibles en ligne. Actuellement, les plus usuelles sont celles proposées par Kolisch [56]. Cependant, il en existe bien d’autres telles que les PACK proposées par Carlier et Néron [27], les BL proposées par Baptiste et Le Pape [10], et les instances de Flow-shop hybrides proposées également par Carlier et Néron
Le paragraphe 5.2.1 du chapitre 5 est consacré à la description détaillée de quelques-unes de ces instances.

1.6 Applications industrielles

Le RCPSP est par définition le problème d’ordonnancement situé au cœur de la gestion de projet et à ce titre possède des applications industrielles directes, notamment dans les secteurs de la construction et des services.

Il existe de nombreuses applications plus inattendues du RCPSP dans le secteur industriel, parmi lesquelles on peut citer, dans le cadre des problèmes mono-ressource, le problème d’atelier de textile consistant à faire des découpes de formes rectangulaires dans la bande de largeur fixée de tissu, de manière à minimiser la longueur du tissu utilisé. Par analogie, la largeur de tissu correspond à la ressource et le makespan correspond à sa longueur. Ce type d’exemple se retrouve également dans l’industrie métallurgique concernant la découpe de tôles.

Dans l’industrie chimique, le processus de production peut être simplifié en un LCSP (labor-constrained scheduling problem), qui consiste à produire des quantités données de produits chimiques, où, à chaque produit correspond une suite d’activités identiques dont les consommations varient en fonction du temps, avec des ressources (travailleurs) disponibles en quantité limitée. En réduisant ce problème au LCSP, le problème lié au processus de production chimique s’apparente aussi au RCPSP.

Étant donné qu'on peut considérer tous les problèmes d'ateliers (Flow-shop, Job-shop, Open-shop ...) comme des cas particuliers du RCPSP, où les activités sont les opérations (liées à un travail) à exécuter sur une ressource unitaire (machine), les applications industrielles de ces problèmes d'atelier restent également valables pour le RCPSP.

Dans le domaine organisationnel, le problème de gestion des emplois du temps consistant en l'organisation d'un certain nombre d'activités faisant appel à différents groupes de personnes, nécessitant du matériel et/ou de l'espace en tenant compte du temps, peut également être assimilé au RCPSP. La planification du personnel dans une compagnie aérienne et la conception d'emplois du temps dans une université, en sont des illustrations pratiques.

Il existe aussi des exemples plus récents comme dans le contexte de l'informatique parallèle sur systèmes centralisés, où des requêtes d'utilisateurs sont effectuées, impliquant l'exécution de jobs à planifier et à synchroniser sur plusieurs processeurs, et/ou sur des ressources limitées en mémoire.

Enfin, dans le contexte industriel régional, comme exemple d'application potentielle du RCPSP, on peut citer le cycle complet conception/production d'un nouvel avion chez Airbus [67].

1.7 Méthodes de résolution des problèmes d'optimisation combinatoire et d'ordonnancement

La résolution de ce problème très complexe, passe par l'exploration de méthodes s'appuyant sur des techniques de résolution diverses dont :

- *La programmation dynamique* : Introduite par Richard Bellman dans les années 1940, la programmation dynamique permet de résoudre tout problème d'optimisation dont la fonction objectif se décrit comme la somme de fonctions monotones non-décroissantes des variables. Il s'agit d'une méthode exacte consistant à plon-

ger le problème proposé dans un problème plus général, dépendant de paramètres entiers, le problème initial correspondant à une valeur précise de ceux-ci. On essaie alors de résoudre le problème général par récurrence sur ces paramètres entiers. La programmation dynamique utilise le principe d'optimalité que l'on peut énoncer de la façon suivante : dans une séquence optimale de décisions, quelle que soit la première décision prise, les décisions suivantes forment une sous-suite qui est optimale, compte tenu des résultats de la première.

– *La programmation linéaire en nombres entiers (PLNE)* : Plusieurs modèles proposés pour la résolution de problèmes d'optimisation combinatoire et d'ordonnancement utilisent la programmation linéaire, qui constitue l'une des principales pistes pour ce type de problème. Parmi les modèles proposés pour le RCPSP, on peut citer les formulations dites indexées par le temps, et les formulations basées sur une énumération des sous-ensembles d'activités exécutables en parallèle. Nous consacrons les deux chapitres suivants aux formulations de type PLNE pour le RCPSP.
– *Le "Branch-and-Bound"* : L'algorithme par séparation et évaluation, également appelé selon le terme anglo-saxon "branch-and-bound", est une méthode générique de résolution de problèmes d'optimisation, et plus particulièrement d'optimisation combinatoire ou discrète. C'est une méthode d'énumération implicite à l'aide d'une arborescence : toutes les solutions possibles du problème peuvent être énumérées, mais l'analyse des propriétés du problème permet d'éviter l'énumération de larges classes de mauvaises solutions (des branches sont coupées). Dans un bon algorithme par séparation et évaluation, seules les solutions potentiellement bonnes sont donc énumérées. Notons que les approches basées sur la PLNE utilisent généralement le branch-and-bound.
– *La programmation par contraintes* : Au sein de la programmation par contraintes, la propagation de contraintes est un ensemble de techniques qui, par retrait de valeurs dites *inconsistantes*, c'est-à-dire rendues impossibles pour une ou plusieurs variables du fait de leur interaction avec les autres variables, filtrent ainsi l'espace des solutions admissibles. La programmation par contraintes a souvent été utilisée

pour le RCPSP dans des modèles hybrides basés sur la PLNE. On trouve dans ce cadre des travaux basés sur des raisonnements de type énergétique [68].

– *Le raisonnement énergétique [68]* : En ordonnancement, si l'on considère qu'une énergie correspond au produit d'un temps par une quantité de ressource, le raisonnement énergétique est une technique qui consiste à évaluer l'énergie disponible pour l'exécution d'une activité sur un intervalle de temps, compte tenu de la consommation des autres tâches.

  Grâce à cette évaluation, des bilans énergétiques peuvent mettre en évidence un manque d'énergie sur cet intervalle, d'où des conditions de séquencement entre activités ou des interdictions de localisation d'une activité sur certains intervalles de temps. Le raisonnement énergétique fournit ainsi des règles performantes et bien connues de propagation (paires de disjonction, sélections immédiates, ensembles non postérieurs/non antérieurs).

– *Les heuristiques spécifiques* : Il s'agit de procédures qui, pour la résolution d'un problème, permettent de déterminer, parmi plusieurs méthodes de recherche, celle qui semble la plus prometteuse pour atteindre le but visé. Elles peuvent être exprimées sous formes de critères, de principes ou de méthodes, et formalisent l'intuition que l'on a de la voie à suivre. Elles représentent un compromis entre deux exigences : le besoin de rendre de tels critères simples, et le désir de les voir établir une distinction entre les bons et les mauvais choix.

– *Les métaheuristiques* : Parmi les métaheuristiques, on parle des méthodes dites de recherche locale (méthode tabou, recuit simulé, méthodes d'acceptation à seuil . . .) et des méthodes d'approche évolutive (méthode de recherche dispersée, algorithme de la fourmi, algorithmes génétiques et mémétiques). Durant ces dernières années, plusieurs métaheuristiques ont prouvé leur efficacité pour la résolution de problèmes combinatoires comme les problèmes d'ordonnancement et notamment le RCPSP.

1.8 Méthodes de résolution pour le RCPSP

Compte-tenu de l'intérêt que suscite le RCPSP, il existe une littérature importante concernant sa résolution, qu'on peut regrouper en trois catégories :

- les méthodes de calcul de bornes inférieures ;
- les méthodes de calcul de bornes supérieures (résolution approchée) ;
- les méthodes de résolution exactes.

1.8.1 Le calcul de bornes inférieures

Ces méthodes consistent à autoriser la violation de certaines contraintes du problème ou à la réduction du problème initial à un problème plus simple dont la solution fournit une borne inférieure pour la valeur optimale du problème initial. La relaxation des contraintes de précédences conduit à l'obtention des bornes inférieures basées sur les ressources, comme la *borne énergétique*, tandis que la relaxation des contraintes de ressources donnent des *bornes de chemin critique*. Explicitement, il s'agit donc d'élargir l'espace des solutions en autorisant certaines solutions irréalisables pour le problème initial.

On peut regrouper les travaux sur le calcul de bornes inférieures en deux grands groupes : ceux faisant appel à des techniques telles que la programmation linéaire en nombres entiers et/ou la programmation par contraintes, et ceux qui consistent à réduire le RCPSP en un problème plus simple. Ces deux approches font l'objet des deux paragraphes suivants.

1.8.1.1 Calcul des bornes inférieures par la PLNE et/ou la PPC

La borne inférieure la plus simple et la plus standard, servant parfois pour l'évaluation des solutions optimales, est la valeur du *chemin critique (CP)*. Il s'agit principalement de la résolution du problème sans contraintes de ressources. Cependant, il existe des travaux proposant des versions améliorées de cette borne. Ainsi Stinson *et al.* [91] propose en 1978 d'ajouter à la valeur de la borne, la durée opératoire des activités n'appartenant

pas au chemin critique et qui violent les contraintes de ressources, si elles s'exécutent en même temps que les activités du chemin critique. Cette approche a servi de base à d'autres auteurs [36] qui améliorent encore cette borne.

Mingozzi *et al.* [73] proposent, à partir d'une PLNE basée sur la notion de bloc, une version améliorée de cette borne. Un bloc correspond à un ensemble d'activités qui, exécutées simultanément, ne violent aucune contrainte de précédence, ni de ressource. Ils utilisent ainsi des variables binaires indiquant si un bloc d'activités donné est en cours d'exécution ou pas, à un instant donné. Cependant, le trop grand nombre de variables générées ne favorise pas une résolution à l'optimum. Pour remédier à ce handicap, Knust et Brucker [22], à la suite des travaux de Baar et al. [8], font appel à la technique de génération de colonnes.

Christofides en 1987 [30], à partir de la formulation à temps discret de Pritsker [83], propose une borne inférieure par relaxation continue plus efficace, renforcée par un certain nombre de coupes, en plus de la contrainte de précédence qui est remplacée par une contrainte désagrégée (nous y reviendrons dans le chapitre suivant). Entre autres choses, il ajoute des coupes de cliques empêchant l'exécution simultanée des activités des ensembles disjonctifs. D'autres auteurs tels que Sankaran *et al.* [88] ont également exploré cette piste.

Il existe des travaux basés sur la relaxation lagrangienne appliquée à la formulation sur les ensembles admissibles de Mingozzi *et al* [73] où le sous-problème devient un problème d'ordonnancement sans contraintes de ressources, mais avec des fenêtres de temps [34]. Lorsque la relaxation lagrangienne est appliquée à la formulation proposée par Brucker et Knust [22], le sous-problème lagrangien se décompose en plusieurs petits problèmes de *sac-à-dos multidimensionnel*.

Combinant la PPC et la PLNE, et intégrant des techniques de "lifting" et des déductions par "shaving" (opérations globales) sur les séquencements des activités pour générer des

inégalités valides, Demassey et al. [34] calculent des bornes pour trois types de formulation : Formulation à temps discret proposée par Christofides, le modèle à temps continu basé sur les flots [7] et le modèle préemptif basé sur les ensembles critiques.

1.8.1.2 Reduction à un problème plus simple

Il existe également des relaxations qui consistent à réduire le RCPSP à un autre problème d'une complexité moindre. Dans cette logique, on trouve la relaxation du RCPSP en un problème à *m* machines par Carlier et Latapie [25]. Chaque ressource est considérée séparément. Le RCPSP peut aussi être assimilé à un problème cumulatif (CuSP) préemptif et multi-élastique en relâchant toutes les contraintes de ressources sauf une [76].

Une relaxation du modèle des ensembles admissibles de Mingozzi *et al* [73] conduit à un problème de "weighted node packing". Cette borne a été intégrée dans la méthode exacte de Demeulemeester et Herroelen [36] qui figure parmi les plus efficaces pour résoudre le RCPSP.
Une autre manière de calculer des bornes inférieures pour le RCPSP consiste à relâcher les contraintes de non-préemption des activités, autorisant ainsi l'exécution morcelée des activités [31].

La méthode de Laborie [60] que nous décrivons plus loin dans la catégorie des méthodes exactes permet également d'obtenir des bornes inférieures de très bonne qualité.
Sachant que l'objectif principal pour cette thèse ne porte pas sur le calcul des bornes inférieures, nous ne détaillerons pas davantage cette partie.

Nous renvoyons le lecteur à l'état de l'art [77].

1.8.2 Les méthodes de résolution approchées : heuristiques et métaheuristiques

Ces méthodes sacrifient le caractère optimal de la solution pour obtenir, en un temps de calcul raisonnable, des solutions sous-optimales de bonne qualité. Ces méthodes reposent généralement sur un mécanisme de déplacement (aléatoire ou non) dans l'espace des solutions. Elles ne sont pas exactes, mais permettent en général d'obtenir des solutions proches de l'optimum.

1.8.2.1 Les heuristiques

À l'instar des travaux sur le calcul des bornes inférieures, il existe de nombreux travaux sur les méthodes de résolution approchées appliquées au RCPSP. Certaines d'entre elles s'appuient sur des schémas classiques de métaheuristiques (méthode tabou, exploration de grands voisinages), tandis que d'autres exploitent des formulations spécifiques au RCPSP. Cependant, compte-tenu du nombre de travaux entrant dans ce cadre, sans aucune prétention d'être exhaustif, nous ne présentons que quelques-uns de ces travaux.

En général, une des difficultés dans un processus heuristique consiste à choisir entre plusieurs solutions. Afin de contourner cette difficulté, une pratique courante est d'utiliser une *liste de priorité* pour les activités. Cette combinaison renforce donc l'efficacité des heuristiques et se retrouve le plus souvent dans des méthodes de résolution approchées telles les algorithmes de liste. Dans ces méthodes, la liste sert d'élément de base à l'heuristique qui définit alors une solution (un ordonnancement) réalisable.

Les algorithmes de liste : ils consistent à construire une solution itérativement de manière gloutonne. À partir d'une règle de priorité choisie, on établit une liste de priorité contenant toutes les activités du problème. Par ordre de priorité dans cette liste, les activités sont ordonnancées en tenant compte de la capacité des ressources. A chaque étape intermédiaire on a donc un *ordonnancement partiel*. On distingue deux algorithmes de liste principaux pour le RCPSP.

Dans un *ordonnancement de liste parallèle*, on construit la solution dans un ordre

chronologique à partir de la date de début du projet ($t = 0$), et à chaque date, on planifie le maximum d'activités possibles ; puis on passe à la date suivante. La solution obtenue fait partie de l'ensemble des ordonnancements sans-délai qui ne contient pas nécessairement la solution optimale. A l'opposé, un *ordonnancement de liste en série*, raisonnant activité par activité, choisit chaque activité dans l'ordre de la liste de priorité et la planifie au plus tôt compte etnu des tâches déjà ordonnancées. L'ordonnancement ainsi obtenu appartient à la classe des ordonnancements actifs qui contient au moins un ordonnancement optimal. Il existe de nombreuses règles de priorité dans la littérature [55], parmi lesquelles on peut citer : l'ordre lexicographique, la date de début au plus tôt, un ordre aléatoire....
Möhring *et al.* [74] proposent un algorithme de liste spécifique à partir d'une solution obtenue par relaxation lagrangienne, lui donnant ainsi une liste de priorité. Cela leur permet de construire une solution réalisable pour le problème.

Recherche locale par voisinages : il s'agit de méthodes itératives qui, à partir d'une solution initiale calculée par une heuristique, applique à chaque itération une modification locale (opération de déplacement vers un voisin) de la solution courante. L'ensemble des solutions (voisins) pouvant être sélectionnées à partir de la solution courante x est appelé *le voisinage de x*. Dans ces méthodes, la définition de voisinage est donc une étape importante.

Recherche tabou : certains auteurs [42] ont utilisé la méthode tabou basée sur un algorithme d'ordre strict pour calculer des solutions approchées pour le RCPSP. L'algorithme d'ordre strict consiste à construire des ordonnancements réalisables à partir d'une liste ordonnée d'activités respectant rigoureusement les contraintes de précédence. De ce fait, on n'ordonnancera une activité que si toutes les activités prédécesseurs l'ont été. Les solutions proposées sont des ordonnancements actifs ou semi-actifs. L'amélioration de la solution peut se faire selon plusieurs stratégies. À la différence de l'exploration totale du voisinage de la solution courante, il peut s'agir, par exemple à partir des fenêtres de temps, d'identifier et d'explorer les éléments du voisinage demeurant réalisables. Les

résultats de ces tests sur les instances classiques sont encourageants.

LNS : Palpant *et al.* [79] développent une méthode de grand voisinage, qui consiste à générer des sous-problèmes successifs du problème initial et à les résoudre en utilisant un solveur commercial tel que ILOG-Scheduler. À chaque itération de cette méthode, on fixe la date de début d'un certain nombre d'activités dans la solution courante. À partir des activités libres (variables de date de début non fixées), on forme un sous-problème où l'on cherche à minimiser la date de fin du projet en ordonnançant les activités libres, en tenant compte des contraintes engendrées par les variables fixées, des fenêtres de temps et des disponibilités des ressources. On injecte la solution obtenue du sous-problème dans la solution courante et on optimise le tout par des heuristiques de type forward-backward. Les résultats de cette méthode affichent de bons résultats en termes de qualité de la solution et de temps de résolution.

Il existe bien d'autres méthodes approchées pour le RCPSP, faisant appel à des techniques diverses. Certaines d'entre elles utilisant la PLNE sont basées sur la formulation même du problème. Nous avons par exemple les heuristiques utilisant la solution de relaxations basées sur les ensembles interdits [39]. Jean Damay *et al.* [32] proposent une approche basée sur l'utilisation des anti-chaînes valides (ensemble d'activités pouvant être exécutées simultanément au cours de l'ordonnancement). Alvarés-Valdez et Tamarit [3] proposent une heuristique basée sur les ensembles interdits minimaux (le plus petit ensemble d'activités ne pouvant pas être exécutées en parallèle). Nous détaillons la notion d'ensemble interdit dans le chapitre suivant. Les auteurs génèrent, à chaque étape de la résolution, une nouvelle contrainte de précédence entre deux activités d'un ensemble interdit minimal. Pour ce faire, ils utilisent diverses stratégies de sélection de ces ensembles, telles que la plus petite cardinalité.

1.8.2.2 Les métaheuristiques

Nous décrivons brièvement quelques approches de résolution, basée sur des métaheuristiques telles que le recuit simulé, la recherche tabou et les métaheuristiques basées

sur les populations.

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. Appliquée à l'optimisation, elle permet de trouver les extrema d'une fonction.
Au début de l'algorithme, une première solution est choisie. Ensuite, un voisin est créé à partir de cette solution. Si le voisin a une meilleure valeur que celle de la solution courante, alors il est accepté. Par contre, s'il n'améliore pas la solution actuelle, il peut être accepté selon une certaine probabilité calculée à partir du rapport suivant : E/T, où E est la différence absolue entre les valeurs de la solution courante et celle du voisin, et T est la température actuelle du système. Lorsque 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, comme le processus converge, la température baisse, donc seules les petites détériorations de la valeur de la fonction-objectif auront une chance d'être tolérées.
Boctor [17] et Bouleimen et Lecocq [18] (la meilleure méthode à ce jour pour la résolution du RCPSP utilisant cette technique) proposent un schéma d'ordonnancement sériel avec une liste de priorité dans laquelle les voisins sont choisis à l'aide d'un opérateur de déplacement dans la liste. D'autres auteurs (Lee et Kim [64] et Cho et Kim [29]) décrivent une méthode de recuit simulé basé sur déplacement aléatoire, dans un schéma d'ordonnancement parallèle.

La *recherche tabou* est également une méthode d'exploration du voisinage dans laquelle on mémorise dans une liste (tabou) un certain nombres de solutions (voisins) testées, évitant ainsi de cycler dans la génération des voisins.
L'application de cette méthode au RCPSP a été réalisée par plusieurs auteurs : Pinson *et al.* [81] en 1994 qui ont été parmi les premiers à l'avoir implémentée, Kochtetov et Stolyar [54] en 2003 appliquent un algorithme de recherche locale avec un voisinage variable, Valls *et al.* [95] proposent une recherche tabou avec utilisation d'un ordre to-

pologique, Artigues *et al.* [7] proposent une méthode de recherche tabou en utilisant un voisinage basé sur la réinsertion dans le réseau de flux de ressources.

Il existe d'autres approches dites *métaheuristiques basées sur des populations*. Principalement, dans ces méthodes, à chaque étape du processus, de nouvelles solutions candidates sont générées, parmi lesquelles certaines seront sélectionnées pour générer une nouvelle population. De façon générale, il s'agit de méthodes assez efficaces pour le RCPSP. Dans cette optique, on peut citer entre autres : Hartmann [44] avec un algorithme génétique, ou Merkle *et al.* [72] qui proposent une optimisation par colonies de fourmis associée à une recherche locale.

1.8.3 Les méthodes de résolution exactes

Ces méthodes garantissent la complétude de la résolution et le caractère optimal des solutions trouvées, mais sont coûteuses en temps de calcul. Elles se caractérisent par une exploration déterministe de l'espace de solutions. On peut regrouper les travaux sur la résolution exacte du RCPSP en trois grandes familles : les travaux basés sur un branch-and-bound, ceux utilisant la programmation par contraintes (PPC) et enfin ceux faisant appel à la programmation linéaire en nombres entiers (PLNE). Cependant, dans la pratique, on retrouve assez souvent des propositions de modèles combinant la PPC avec la PLNE ou utilisant la PPC en prétraitement dans les branch-and-bound.

1.8.3.1 Les méthodes de type branch-and-bound

Les méthodes de type Branch-and-Bound construisent un arbre de recherche afin d'explorer implicitement l'espace de recherche. A chaque nœud de l'arbre de recherche, deux situations se produisent :

- un nœud feuille est atteint et une solution économiquement viable, peut être déduite ou bien après le calcul de bornes inférieures, on atteint une borne supérieure ce qui dans les deux cas stoppe l'exploration à partir du nœud ;
- on se retrouve à un nœud (nœud courant) permettant de diviser l'espace de re-

cherche associé en des sous-ensembles tels que leur union corresponde à l'ensemble des solutions du nœud courant. Cette opération de partitionnement d'un espace de recherche en sous-zones s'appelle la séparation ou le branchement.

Construire un arbre de recherche qui explore de manière efficace toutes les solutions pertinentes est un défi du problème. Il est connu que l'efficacité d'un système de branchement est généralement liée à la fois à la qualité des bornes inférieures et aux règles de dominance appliquées. Déjà dans les premiers travaux sur le RCPSP [48], de nombreux auteurs ont proposé plusieurs schémas de branchement dans la littérature. Il existe principalement deux approches concernant les types de branchement. La première consiste à ajouter une seule activité à chaque nœud, la seconde intègre tout un sous-ensemble d'activités.

Branchement d'une activité par nœud

Patterson *et al.* en 1990 [80] et Sprecher en 2000 [89] proposent tous deux des schémas de branch-and-bound basés sur des systèmes de branchement chronologiques où, à chaque nœud (correspondant ici à un ordonnancement partiel actif ou semi-actif (cf. §1.2), on ajoute une activité dite *admissible* (activité dont tous les prédécesseurs ont été ordonnancés) tout en respectant les contraintes de ressource et de précédence. Dans cette logique, Baptiste *et al.* [11] proposent de choisir à chaque nœud l'activité admissible ayant la plus petite date de début au plus tôt.
Sprecher [89] propose la règle de dominance (règle du déplacement global à gauche) étendant la notion d'ordonnancement actif. Cette règle consiste à caler chaque activité le plus à gauche possible, à chaque nœud de l'arbre.

Branchement d'un sous-ensemble d'activités par nœud

Cette approche initiée par Christofides *et al.* [30] a été grandement affinée par Demeulemeester and Herroelen [36]. Notons que les méthodes de type branch-and-bound fondées sur ces schémas de branchement sont parmi les plus efficaces sur les instances classiques de la littérature. Cette approche est basée sur la notion de *"Minimal Delaying*

Alternatives" (MDA).

Un MDA est lié au concept d'ensemble interdit et au concept d'ensemble réalisable maximal. Un ensemble interdit E est un ensemble d'activités reliées entre elles par aucune relation de précédence, mais ne pouvant pas être exécutées toutes en parallèle sans provoquer un conflit de ressource. Un ensemble réalisable maximal est un ensemble d'activités réalisables simultanément et maximal au sens de l'inclusion. Etant donné un ensemble interdit E, un MDA est un sous-ensemble M de E tel que $E \setminus M$ soit un ensemble réalisable maximal. C'est en fait un sous-ensemble d'activités minimal qu'il faut décaler pour pouvoir exécuter les autres activités simultanément. A chaque nœud, s'il n'y a aucun conflit de ressource, on ordonnance toutes les activités admissibles, et on crée le seul nœud suivant. Dans le cas contraire, on énumère et on crée un nœud pour chaque MDA.

À l'opposé de la méthode proposée ci-dessus, une autre manière de brancher proposée par Mingozzi *et al.* [73], est de séparer chaque nœud correspondant à un ensemble réalisable.

Il existe bien d'autres approches de branchement dans la littérature telle que la proposition de Brucker *et al.* [23] basée sur la fixation des paires de disjonction et des activités pouvant être exécutées en parallèle. Laborie [60] propose une méthode destructive basée sur le concept d'ensemble critique minimal *(minimal critical sets (MCS))*, une autre désignation des *ensembles interdits minimaux* de Avalrés-Valdez et Tamarit.

1.8.4 Les modèles utilisant la programmation par contraintes (PPC)

La programmation par contraintes est d'un très grand intérêt pour la résolution du RCPSP. Dans le cadre d'une résolution exacte, sa démarche consiste à décomposer le problème en éléments simples, c'est-à-dire des contraintes qui interagissent entre elles. À travers ses techniques de propagation de contraintes, elle permet notamment la réduction des fenêtres de temps des activités. Utilisée très souvent en prétraitement (preproces-

sing), elle permet (par exemple grâce au raisonnement énergétique et la recherche d'inconsistance globale des contraintes) de tester la réalisabilité du problème et même d'affiner le contour du polyèdre représentant l'espace de solutions [11]. De ce fait, elle réduit la taille de l'arbre en éliminant certains nœuds moins intéressants à explorer [6, 21]. Pour plus de détails, voir [11]

Cette approche fournit des résultats intéressants, mais elle est mal adaptée pour une exploration complète de l'espace de recherche. C'est pourquoi elle est très souvent utilisée en coopération avec la programmation linéaire en nombres entiers.

1.8.5 Les modèles utilisant la programmation linéaire en nombres entiers (PLNE)

La PLNE, depuis les premiers travaux sur le RCPSP [9, 83], demeure l'une des pistes les plus explorées pour la résolution de ce problème. Nous y consacrons le prochain chapitre.

1.9 Quelques extensions du RCPSP

Du fait de son très grand nombre d'applications possibles dans le domaine industriel, il existe de même de nombreuses extensions du RCPSP. Ces extensions découlent essentiellement de la variation de certaines données du problème, de la généralisation de certaines contraintes ou tout simplement de l'optimisation du problème à partir de nouveaux critères autres que les critères classiques dont le $C_{\max}$. Ainsi, également sans aucune prétention d'être exhaustif, nous n'en citerons que quelques-unes.

RCPSP préemptif : Il s'agit d'une variante du RCPSP autorisant la préemption des activités ; c'est-à-dire qu'on peut interrompre l'exécution d'une activité pour en débuter (ou poursuivre ou terminer) une autre. Cette variante correspond à une relaxation du cas non-préemptif et sa solution optimale est considérée comme une borne inférieure du makespan dans le cas non-préemptif.

RCPSP avec contraintes de précédence généralisées : Dans cette variante, on généralise les relations de précédence entre les activités. Ainsi, les contraintes de précédence sont modélisées de la manière suivante :

$$S_j \geq S_i + l_{i,j} \quad \forall (i,j) \in A^2, \tag{1.5}$$

où $l_{i,j}$ peut prendre n'importe quelles valeurs positives, négatives ou nulles.

On parlera de *RCPSP Multi-Mode* lorsqu'une activité pourra s'exécuter selon plusieurs modes. En outre, d'un mode à l'autre, la durée et les consommations de ressources de l'activité peuvent varier. En pratique, la résolution de ce type de problème consiste en un premier temps, à définir le mode d'exécution de chaque tâche, puis à trouver la solution optimale en fonction des choix faits.

Il existe également certains travaux sur la *robustesse* et la *flexibilité* en RCPSP, ainsi que d'autres extensions s'intéressant à la *prise en compte des incertitudes* telles que l'insertion des activités imprévues ou d'une ressource supplémentaire. En plus de travaux sur les *formulations de type stochastique du RCPSP* et des extensions de type *Multi-Projet* (gestion de plusieurs projets en parallèle), il existe aussi des travaux sur le *RCPSP avec les consommations et les disponibilités* des activités qui varient. D'autres travaux font appel aussi à des *ressources non-renouvelables* (nous y reviendrons dans le chapitre 4 dans le cas du RCPSP avec consommation et production de ressources) ou *doublement contraintes*, des critères (fonctions-objectifs) différents du makespan tels que des *fonctions de coût spécifique* (par exemple, la valeur actuelle nette du projet).

1.10 Conclusion

Dans ce chapitre, nous avons présenté le problème d'ordonnancement de projet sous contraintes de ressources (RCPSP). Ainsi, après une description détaillée présentant un exemple et certaines applications industrielles, nous avons évoqué quelques techniques et travaux de résolution (état de l'art) qu'on peut trouver dans l'abondante littérature qui

lui a été consacrée. Enfin, nous avons présenté brièvement quelques extensions de ce problème.

Le prochain chapitre (chapitre 2) est consacré à la résolution du RCPSP par la programmation linéaire en nombres entiers (PLNE).

CHAPITRE 2

UTILISATION DE LA PLNE POUR LE RCPSP

De nombreux modèles de programmation linéaire en nombres entiers (PLNE) ou mixtes sont proposés dans la littérature pour la résolution du RCPSP . On distingue notamment :

- les formulations dites *étendues*, comportant un nombre *exponentiel* de variables telles que la formulation à temps discret de Mingozzi et al. [73], et celle à temps continu d'Alvarez et al. [3].
- les formulations dites *compactes*, comportant un nombre *polynomial* de variables,
- les formulations qui comportent un nombre *pseudo-polynomial* de variables (les formulations à temps discret de Pritsker et al. [83] et de Christofides et al. [30])

Ainsi, dans ce chapitre portant particulièrement sur les formulations de type programme linéaire en nombres entiers ou mixtes proposées pour la résolution du RCPSP, nous présenterons de façon succincte et sans être exhaustif, quelques formulations issues de la littérature. Trois de ces formulations seront étudiées plus finement. Il s'agit de deux formulations à temps discret (la formulation basique à temps discret [83] et la formulation désagrégée à temps discret [30]), ainsi qu'une formulation à temps continu (basée sur les flots [7]).

2.1 La programmation linéaire en nombres entiers

Un *programme linéaire (PL)* [96] est un problème d'optimisation qui consiste à maximiser ou minimiser une fonction-objectif linéaire soumise à un ensemble de contraintes exprimées sous la forme d'équations ou d'inéquations linéaires. Ces contraintes définissent le polyèdre des solutions admissibles. La principale caractéristique d'un programme linéaire réside dans sa facilité de résolution grâce au célèbre algorithme de résolution du Simplexe, qui consiste à se déplacer d'un sommet du polyèdre des solutions admissibles à un sommet adjacent de coût inférieur (ou égal). Ainsi, l'algorithme

évalue de façon itérative différentes solutions du problème afin d'en extraire une de coût optimal. Lorsque les variables sont de plus contraintes à être entières, on parle de *programme linéaire en nombres entiers (PLNE)*. Les PLNE sont difficiles à résoudre, du fait notamment que le domaine admissible n'est plus convexe mais discret. Notons néanmoins que la programmation linéaire en nombres entiers est un outil très utile de modélisation de problèmes comportant des contraintes logiques, des discontinuités et bien d'autres aspects combinatoires. De ce fait, de nombreux problèmes d'optimisation issus de domaines d'applications très divers, sont modélisés comme des PLNE.
Un PLNE dans sa forme la plus générale se présente de la manière suivante :

$$\max_x z := cx$$
$$ax \leq b$$
$$x \in \mathbb{N}^n$$

où a est une matrice $n \times m$, b est un vecteur colonne de taille m, c et b sont des vecteurs (respectivement ligne et colonne) de taille n. Les composantes du vecteur colonne x (de taille n) sont des variables d'optimisation du problème.

Différentes approches ont été étudiées pour la recherche de solutions optimales entières des PLNE. Les plus performantes font essentiellement appel à des algorithmes de recherche arborescente par séparation et évaluation (*branch and bound*) et à des méthodes de coupes.

L'idée de la *recherche arborescente par séparation et évaluation* est d'effectuer une énumération implicite des solutions du PLNE. Cette méthode consiste à séparer le problème en plusieurs sous-problèmes, chacun avec un domaine admissible qui est un sous-ensemble de domaine admissible du problème, puis à évaluer chaque sous-problème en calculant une borne supérieure de la valeur optimale dans le cas d'un problème de maximisation (ou une borne inférieure de sa valeur optimale dans le cas d'un problème de minimisation). Cette borne est ensuite comparée à une valeur de référence correspondant

à une solution connue du problème. On réitère ensuite le processus uniquement sur les sous-problèmes dont l'évaluation est supérieure (respectivement inférieure) à la valeur de référence dans le cas de la maximisation (resp. minimisation), car ils sont les seuls susceptibles de l'améliorer. L'ensemble des solutions admissibles est ainsi représenté par une arborescence dans laquelle un grand nombre de nœuds peuvent être éliminés. L'évaluation des sous-problèmes se fait typiquement par *relaxation linéaire* (on ignore la contrainte d'intégrité) en utilisant le Simplexe.

Les méthodes de coupes ont pour but (idéalement) de trouver l'enveloppe convexe des solutions admissible (entières), c'est-à-dire le plus petit polyèdre contenant toutes les solutions admissibles (entières) du PLNE. Si l'on arrive à déterminer cette enveloppe convexe, la résolution de la relaxation du PLNE réduit à cet ensemble donne une solution optimale entière. La difficulté de ces méthodes réside dans la génération de coupes efficaces. Les méthodes de coupes utilisées seules affichent des performances modestes. En revanche, elles se révèlent efficaces lorsqu'associées à des méthodes de recherche arborescente telles que le *branch and cut* [96] (méthode combinant l'algorithme du Branch and Bound et la méthode des coupes polyédrales).

2.2 Résolution du RCPSP par la PLNE

Les travaux les plus anciens menés sur la résolution exacte du RCPSP font appel à la programmation linéaire en nombres entiers. Dans cette approche, du fait qu'il est nécessaire de garder une trace de l'ensemble des activités en cours d'exécution à chaque instant, les contraintes de ressources sont les plus difficiles à modéliser au moyen d'inégalités linéaires (cf. contrainte (1.2), §1.2, du chapitre précédent). Pour y remédier, une solution proposée consiste en la discrétisation du temps au détriment d'une augmentation du nombre de variables, dépendant alors de la valeur de l'horizon d'ordonnancement (cas des formulations de Pritsker *et al.* [83] et Christofides *et al.* [30]). En général, ces modèles possèdent soit un nombre exponentiel, soit un nombre pseudo-polynomial de variables. Une autre solution proposée consiste à utiliser des variables réelles pour de-

terminer les dates de début des activités et des variables binaires de séquencement pour modéliser les relations de précédence entre les activités. Même si les modèles basés sur cette approche possèdent un nombre polynomial de variables, ils présentent en général de moins bonnes relaxations linéaires. Cela est du à la présence de coefficients de grande valeur servant à annuler certaines contraintes en fonction des valeurs prises par les variables séquentielles (coefficients dits “big-M”) qui provoquent l’obtention de solutions hautement fractionnaires

2.3 Les formulations à temps discret

Basées sur l’utilisation de variables indexées sur un horizon de temps discrétisé, les formulations à temps discret ont donné lieu à la formulation basique à temps discret (DT) et la formulation désagrégée à temps discret (DDT).

2.3.1 Formulation basique à temps discret (DT)

Proposée en 1969 par Pritsker *et al.* [83], la formulation basique à temps discret est basée sur l’utilisation pour chaque activité i de variables d’optimisation binaire x_{it} indexées par le temps et la discrétisation de l’horizon d’ordonnancement. Précisément, il s’agit d’un découpage de l’horizon en intervalles de temps constants et suffisamment petits, afin de pouvoir représenter toutes les durées opératoires sans les fractionner. La valeur de l’intervalle peut être fixée par exemple au plus petit commun diviseur des différentes durées opératoires, ou à la valeur unitaire de l’unité de mesure du temps utilisée pour estimer les durées opératoires des activités du projet (heures, jours, semaines, mois . . .). La figure 1.2 du chapitre précédent donne un exemple de diagramme de Gantt avec un horizon de temps discrétisé, sur lequel des activités sont ordonnancées.

Pour la suite, on définit une variable binaire x_{it} indexée sur cet horizon de temps discrétisé, telle que x_{it} vaut 1 si l’activité i démarre son exécution à l’instant t, et vaut 0 pour tous les autres moments de l’horizon de temps. Elle permet donc de déterminer sans ambiguïté la date de début d’exécution de chaque activité.

2.3.1.1 Formulation

Á la suite des indications données ci-dessus, on peut alors formuler ce modèle, en utilisant la notation du paragraphe précédent, sous la forme du programme linéaire en nombres entiers suivant :

$$\min \sum_{t \in H} t x_{n+1,t} \tag{2.1}$$

$$\sum_{t \in H} t x_{jt} \geq \sum_{t \in H} t x_{it} + p_i \qquad \forall (i,j) \in E \tag{2.2}$$

$$\sum_{i=1}^{n} b_{ik} \sum_{\tau = t - p_i + 1}^{t} x_{i\tau} \leq B_k \qquad \forall t \in H, \forall k \in R \tag{2.3}$$

$$\sum_{t \in H} x_{it} = 1 \qquad \forall i \in A \cup \{0, n+1\} \tag{2.4}$$

$$x_{it} \in \{0,1\} \qquad \forall i \in A \cup \{0, n+1\}, \forall t \in H, \tag{2.5}$$

2.3.1.2 Description des contraintes

La fonction-objectif est donnée par l'équation (2.1). Elle consiste à minimiser la date de début de l'activité fictive de fin de projet $n+1$, sachant que celle-ci est l'activité jalon de durée opératoire nulle ($p_{n+1} = 0$), marquant la fin du projet. Par conséquent, chercher à minimiser cette date de début, revient donc à minimiser la date de fin du projet (*makespan*).

La minimisation de cette fonction-objectif est soumise aux contraintes suivantes :

– la contrainte (2.2) représente la contrainte de précédence ; elle modélise les relations de précédences entre les différentes activités, en stipulant que si la paire d'activités (i, j) appartient à l'ensemble des précédences E, alors la date de début de l'activité j sera ultérieure ou égale à la date de début l'activité i, augmentée de la durée opératoire de l'activité i (cf. la proposition 2.3.1 qui suit).

– la contrainte (2.3) représente la contrainte de ressource ; elle est aussi appelée *contrainte cumulative*, elle s'assure, pour chaque ressource que la somme des consommations des activités en cours d'exécution ne dépasse pas la capacité de la

ressource.

– La *contrainte de non-préemption* des activités, modélisée par l'équation (2.4), impose à chaque activité de ne débuter qu'une et une seule fois, sur toute la durée du projet. Ceci revient à interdire qu'on puisse interrompre (*préempter*) une activité déjà en cours d'exécution pour débuter/poursuivre/achever une autre.

Proposition 2.3.1. *La formulation conceptuelle (présentée dans la chapitre précédent) est équivalente à la formulation DT.*

Démonstration. Si $(i, j) \in E$, et si l'activité i démarre au temps t, si et seulement si $x_{it} = 1$. Ainsi, $\sum_{t \in H} t x_{it} = t$ en vertu des contraintes (2.4) et (2.5). On a alors : $\sum_{t \in H} t x_{jt} \geq \sum_{t \in H} t x_{it} + p_i \Longleftrightarrow \sum_{t \in H} t x_{jt} \geq t + p_i$. Ce qui revient à dire que pour tout $t' \in H, \quad x_{jt'} = 0$ si $t' \leq t + p_i$. En d'autres termes, l'activité j ne peut débuter qu'après l'achèvement de l'activité i. Les contraintes de ressource de cette formulation sont aussi équivalentes à celles de la formulation conceptuelle. □

En outre, on retrouve aisément la formulation conceptuelle du RCPSP (présentée dans le chapitre précédent) à travers ce modèle, en traduisant les contraintes de précédence (2.2), de non-préemption (2.4) et la définition de l'objectif (2.1) grâce à la correspondance $S_i = \sum_{t \in H} x_{it}$.

2.3.1.3 Caractéristiques du modèle

Ce modèle, caractérisé par l'utilisation de variables d'optimisation binaires indexées sur le temps, possède $(n+2)(T+1)$ variables et $|E|+(T+1)m$ contraintes. Ces nombres pouvant évoluer très rapidement avec l'horizon de temps, ce modèle est classé parmi les modèles comprenant un nombre pseudo-polynomial de variables et de contraintes. Cet aspect peut rapidement s'avérer pénalisant pour ce modèle pour la résolution des problèmes à horizon de temps très étendu. Cependant, cette formulation est connue pour fournir d'assez bons résultats sur la résolution des instances classiques du RCPSP. Elle produit aussi des bornes intéressantes par relaxation continue.

2.3.2 Formulation désagrégée à temps discret (DDT)

La formulation DDT proposée par Christofides en 1987 [30] est très proche de celle proposée par Pritsker (DT). Elle utilise les mêmes variables d'optimisation binaires indexées sur le temps. Cependant, même si la fonction-objectif, la contrainte de ressource et la contrainte de non-préemption restent identiques, la contrainte de précédence de la formulation DDT diffère de celle proposée dans la formulation DT. Ainsi, à la différence de la formulation DT, qui définit la contrainte (2.2) pour chaque paire d'activités de l'ensemble des précédences, la formulation DDT propose une contrainte non seulement pour chaque paire d'activités de l'ensemble des précédences, mais aussi pour chaque instant de l'horizon d'ordonnancement :

$$\sum_{\tau=t}^{T} x_{i\tau} + \sum_{\tau=0}^{t+p_i-1} x_{j\tau} \leq 1, \quad \forall t \in H, \forall (i,j) \in E,$$

La formulation DDT est donc :

$$\min \sum_{t \in H} t x_{n+1,t} \tag{2.1}$$

$$\sum_{\tau=t}^{T} x_{i\tau} + \sum_{\tau=0}^{t+p_i-1} x_{j\tau} \leq 1, \qquad \forall t \in H, \forall (i,j) \in E, \tag{2.6}$$

$$\sum_{i=1}^{n} b_{ik} \sum_{\tau=t-p_i+1}^{t} x_{i\tau} \leq B_k \qquad \forall t \in H, \forall k \in R \tag{2.3}$$

$$\sum_{t \in H} x_{it} = 1 \qquad \forall i \in A \cup \{0, n+1\} \tag{2.4}$$

$$x_{it} \in \{0,1\} \qquad \forall i \in A \cup \{0, n+1\}, \forall t \in H, \tag{2.5}$$

2.3.2.1 Description des contraintes

Proposition 2.3.2. *La contrainte (2.6) stipule que si l'activité i qui précède l'activité j ne commence pas avant l'instant t, alors l'activité j ne pourra pas commencer avant l'instant* $t + p_i$.

Démonstration. Considérons un instant donné, noté t. Que l'activité i ne débute qu'après

l'instant t, peut s'écrire : $\sum_{\tau=t}^{T} x_{i\tau} = 1$.
Dans ce cas, une activité j précédée par i, ne pourra débuter qu'après l'instant $t + p_i$, ce qui peut s'écrire de la manière suivante : $\sum_{\tau=0}^{t+p_i-1} x_{j\tau} = 0$.
De ce fait, la relation suivante :
$\forall (i, j) \in E$ et $\forall t \in H$, (i débute après t) $\Longrightarrow$ (j débute après $t + p_i$),
se modélise comme suit :
$((\sum_{\tau=t}^{T} x_{i\tau} = 1) \Longrightarrow (\sum_{\tau=0}^{t+p_i-1} x_{j\tau} = 0))$ ou encore $((\sum_{\tau=t}^{T} x_{i\tau} = 0) \vee (\sum_{\tau=0}^{t+p_i-1} x_{j\tau} = 0))$,
en vertu de la propriété logique : $(A \Longrightarrow B) \Longleftrightarrow ((\neg A) \vee B)$.
de façon équivalente, en utilisant la binarité des variables, on a :
$\sum_{\tau=t}^{T} x_{i\tau} + \sum_{\tau=0}^{t+p_i-1} x_{j\tau} \leq 1, \quad \forall t \in H, \forall (i, j) \in E$, d'où la contrainte (2.6) ci-dessus. □

2.3.2.2 Caractéristiques du modèle

Classée également parmi les formulations comprenant un nombre pseudo-polynomial de variables, cette formulation qui comporte $(n+2)(T+1)$ variables binaires et $(T+1)(m+|E|)$ contraintes est également connue pour avoir de meilleures relaxations continues que la formulation DT (cf. [6]). Ses contraintes de précédence, en nombre plus élevé que dans la formulation DT, lui permettent de définir plus finement l'enveloppe convexe des solutions, et donc de produire de meilleures solutions continues. En effet, on peut montrer que (2.6) et (2.4) $\Rightarrow$ (2.2) même lorsque les variables x_{it} peuvent être fractionnaires Par contre, celles-ci nécessitent davantage de mémoire lors de l'utilisation de ce modèle par le simplexe.
Les travaux sur les formulations à temps discret ne se limitent pas à ces deux formulations. Parmi les autres formulations, citons Kaplan et Klein qui ont développé des formulations similaires à celles définies ci-dessus. Singulièrement, Kaplan [50] définit les variables de décision y_{it} égales à 1 si i est en cours d'exécution au temps t. Ceci permet d'exprimer plus simplement les contraintes de ressources :

$$\sum_{i \in A} y_{it} b_{ik} \leq B_k \qquad \forall k \in R. \tag{2.7}$$

Par contre, cette simplicité ne s'étend pas à la formulation des contraintes pour la durée

des activités, pour les contraintes de non-préemption et pour les contraintes de précédence.
La formulation de Klein [52] utilise les variables de décision $y_{it} = 1$ si $S_i + p_i < t$. Elle simplifie non seulement l'expression des contraintes de ressources comme le modèle de Kaplan, mais aussi les contraintes de non-préemption et de précédence.

Mingozzi *et al.* [73] quant à eux, construisent un modèle sur la notion d'*ensembles admissibles*. Typiquement, un ensemble $F \subseteq A$ d'activités est dit *admissible* si toutes les activités qui le composent peuvent être exécutées simultanément (en respectant les contraintes de précédence et de ressources). Sur la base de la formulation de Pritsker, il s'agit d'ajouter un nombre exponentiel de variables binaires y_{lt} définies pour tout instant t, et pour tout ensemble admissible l.

2.3.3 Les formulations à temps continu

Ces formulations introduisent le type de variables binaires dites *variables séquentielles*, qui permettent de modéliser les relations de précédences entre les activités : x_{ij} est égal à 1 si l'activité i précède l'activité j, 0 sinon. Soit C un sous-ensemble de A. On dit que C est *ensemble critique minimal* de A, s'il est défini formellement par :

$$
\begin{aligned}
&(C \times C) \cap E = \emptyset \\
&\exists k \in R, \quad \sum_{i \in C} b_{ik} > B_k \\
&\forall C' \subseteq C, \forall k \in R, \quad \sum_{i \in C'} b_{ik} \leq B_k
\end{aligned}
$$

Notons que C_m représente l'ensemble de tous les ensembles critiques d'activités minimaux.
En se basant sur les travaux de Balas [9] portant sur la formulation disjonctive du RCPSP, on retrouve une modélisation des contraintes de ressources (cf. contrainte (1.2)) au moyen des *ensembles critiques minimaux* d'activités *minimaux* dans la formulation proposée par Alvarez-Valdès et Tamarit [3]. Suite à la définition ci-dessus, la contrainte

de ressources peut être modélisée de la manière suivante :

$$\sum_{(i,j)\in C^2} x_{ji} \geq 1 \quad \forall C \in C_m \tag{2.8}$$

La contrainte (2.8) à travers les variables de séquencement x_{ji}, interdit d'exécuter en parallèle l'ensemble des activités d'un même ensemble critique minimal C. Le nombre de contraintes (2.8) est exponentiel.

2.3.4 Formulation à temps continu basée sur les flots (FCT)

Toujours inspiré par les travaux de Balas *et al.*, et sur la base de la formulation d'Alvarez-Valdès et Tamarit, Artigues *et al.* [7] proposent l'utilisation des flots pour la gestion des ressources, en lieu et place des ensembles critiques pour obtenir un modèle plus compact. Dans la suite, cette proposition est nommée *formulation à temps continu basée sur les flots (FCT)*. Ce modèle utilise les variables continues usuelles de date de début S_i pour définir la date de début de chaque activité du projet. Dans ce modèle, on utilise également des variables séquentielles binaires x_{ij} permettant de définir un ordre partiel entre les différentes activités. Ainsi, la variable $x_{ij} = 1$, si l'activité i précède l'activité j. La gestion des ressources dans un système de flot, dans ce modèle, passe par l'utilisation des variables continues f_{ijk} (dites *variables de flots*). En termes informels, la variable f_{ijk} définit la quantité de ressource k "envoyée" (redevenue disponible) par l'activité i (à la fin de son exécution) à l'activité j (au début de son exécution). On considère qu'au début du projet les ressources sont toutes disponibles et stockées au niveau de l'activité fictive 0 de début de projet, servant alors de sommet *source* dans le graphe de précédence associé au projet. Par la suite, les activités démarrant leur exécution utilisent ces ressources pour être exécutées, et une fois terminées, elles transfèrent ces ressources aux activités qui les succèdent. Cette opération se répète jusqu'à ce que les dernières activités à être exécutées, les transmettent à leur tour à l'activité fictive $n+1$ de fin de projet, servant de sommet *puits* dans le graphe de précédence.

2.3.5 Formulation

Dans ce modèle, on utilisera aussi des données b_{ik} correspondant à la quantité de ressource k consommée par l'activité i pendant son exécution. On notera $\tilde{b}_{ik} := b_{ik}$ pour tout $i \in A$, quelque soit $k \in R$ et $\tilde{b}_{0k} = \tilde{b}_{n+1,k} := B_k$, pour les activités fictives 0 (source) et $n+1$ (puits).
Le modèle FCT utilise comme variables de décision, des variables binaires (séquentielles) x_{ij}, des variables continues (de flots) f_{ijk} et des variables continues S_i du modèle conceptuel. Soit $\mathscr{A} := A \cup \{0, n+1\})$

$$\min S_{n+1} \tag{2.9}$$

$$x_{ij} + x_{ji} \leq 1, \qquad \forall (i,j) \in \mathscr{A}^2, i < j \tag{2.10}$$

$$x_{ik} \geq x_{ij} + x_{jk} - 1 \qquad \forall (i,j,k) \in \mathscr{A}^3 \tag{2.11}$$

$$S_j - S_i \geq -M_{ij} + (p_i + M_{ij}) x_{ij} \qquad \forall (i,j) \in \mathscr{A}^2 \tag{2.12}$$

$$f_{ijk} \leq \min(\tilde{b}_{ik}, \tilde{b}_{jk}) x_{ij} \qquad \forall (i,j) \in (A \cup \{0\} \times A \cup \{n+1\}), \forall k \in R \tag{2.13}$$

$$\sum_{j \in \mathscr{A}} f_{ijk} = \tilde{b}_{ik} \qquad \forall i \in \mathscr{A}, \forall k \in R \tag{2.14}$$

$$\sum_{i \in \mathscr{A}} f_{ijk} = \tilde{b}_{jk} \qquad \forall i \in \mathscr{A}, \forall k \in R \tag{2.15}$$

$$f_{n+1,0k} = B_k \qquad \forall k \in R \tag{2.16}$$

$$x_{ij} = 1 \wedge x_{ji} = 0 \qquad \forall (i,j) \in TE \tag{2.17}$$

$$f_{ijk} \geq 0 \qquad \forall (i,j) \in \mathscr{A}^2, \forall k \in R \tag{2.18}$$

$$S_0 = 0 \tag{2.19}$$

$$S_i \geq 0 \qquad \forall i \in \mathscr{A} \tag{2.20}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathscr{A}^2 \tag{2.21}$$

où M_{ij} est une constante (suffisamment grande), telle une borne supérieure valide qui joue le rôle d'une constante de type *grand_M*. Elle peut être fixée à n'importe quelle borne supérieure valide de valeur $S_i - S_j$ (par exemple, $M_{ij} = ES_i - LS_j$). TE est la fermeture transitive de l'ensemble de précédence E. Tout comme dans la formulation

conceptuelle du problème, ici, l'objectif consiste à minimiser la date de fin de la dernière activité du projet, c'est-à-dire minimiser S_{n+1} (*makespan*).

2.3.6 Description des contraintes

Dans l'optique d'une interprétation détaillée des contraintes de ce modèle, on constate que :

- Les inéquations (2.10) assurent que pour toute paire d'activités distinctes $(i, j) \in \mathscr{A}^2$, soit i précède j, soit j précède i, soit i et j s'exécutent en parallèle.
- Les contraintes (2.11) garantissent la transitivité entre les activités de même que l'absence de cycles dans le graphe de précédence. Ainsi, elles imposent que pour chaque triplet d'activités (i, j, k) quelconque, si i précède j, qui lui même précède k, alors i précèdera k.
- Les contraintes (2.12) représentent les contraintes de précédence. Pour toute paire d'activités $(i, j) \in E$, elles obligent j à ne démarrer qu'après l'achèvement de i. Elles permettent par la même occasion de lier les valeurs des variables x_{ij} et S_i. Explicitement, si $x_{ij} = 1$ (c'est-à-dire si i précède j), alors $S_j - S_i = p_i$. Par contre, si $x_{ij} = 0$, pour une valeur de M suffisamment grande (par exemple $M = H$, c'est-à-dire M est fixé à la valeur maximale de l'horizon d'ordonnancement), alors la distance $S_j - S_i$ n'est pas contrainte car on aura $S_j - S_i \geq -M$ (contrainte triviale satisfaite par toute solution optimale du problem).
- Les contraintes (2.13) lient les variables de flot f_{ijk} aux variables séquentielles x_{ij} via la disponibilité des ressources. Ainsi, si i précède j, alors le flot maximal de ressources envoyées de i à j correspond au minimum entre la consommation b_{ik} de l'activité prédécesseur i et la consommation b_{jk} de l'activité successeur j, ceci quelque soit la ressource k.
- Les contraintes de ressources (2.14), (2.15) et (2.21) assurent la conservation des flots. Pour chaque activité $i \in A$, la contrainte (2.14) (respectivement (2.15)) fixe la somme des flux de la ressource k envoyés par i vers ses activités successeurs (respectivement reçues de ses activités prédécesseurs), à sa consommation $\tilde{b}_{ik}$ (respectivement $\tilde{b}_{jk}$).

2.3.7 Caractéristiques du modèle

Ce modèle comportant $(n+2)^2$ variables binaires, $m(n+2)^2+n+2$ variables continues, et $(n+2)^3+2(n+2)^2+m(n+1)^2$ contraintes, peut être classé parmi les modèles compacts. Le processus de résolution est basé *a priori* sur la détermination d'un ordre partiel entre les activités donné par les variables séquentielles x_{ij}, et *a posteriori* sur la détermination des dates de début des activités. Dans leur étude expérimentale sur le problème de job-shop (qui peut être assimilé à un cas particulier du RCPSP), Applegate et Cook [4] montrent que cette formulation produit de mauvaises relaxations dues à l'utilisation de la constante grand_M dans les contraintes. Par contre, pour la résolution des problèmes ayant de grands horizons, cette formulation peut être préférable. Cependant, l'utilisation des variables séquentielles pourrait être à l'origine d'un trop grand nombre de symétries, affectant de ce fait les performances du modèle pour la résolution de problèmes fortement cumulatifs.

2.4 Conclusion

Dans ce chapitre, nous avons passé en revue quelques formulations de type PLNE pour le RCPSP. Nous avons étudié plus finement trois de ces formations. Il en ressort que les formulations DT et DDT, utilisant les variables indexées sur le temps (nombre pseudo-polynomial de variables) bien que connues pour donner de bons résultats, ne sont pas adaptées pour les problèmes comportant de grands horizons d'ordonnancement. La formulation FCT de type compact, quant à elle, est mieux adaptée pour les problèmes de grands horizons, mais ne l'est pas pour les problèmes très cumulatifs. Un enjeu important, serait alors de proposer de nouvelles formulations susceptibles de traiter des problèmes à la fois très cumulatifs, mais ayant aussi des horizons de temps importants. Dans le chapitre suivant, nous proposons deux nouvelles formulations conçues dans cette optique.

CHAPITRE 3

FORMULATIONS PLNE BASÉES SUR LES ÉVÉNEMENTS

Le chapitre précédent décrit les formulations de type étendue. Compte tenu de leur expression, on comprend que ce type de formulation résout difficilement des instances associées à des horizons d'ordonnancement importants. D'autre part, la formulation FCT utilisant les variables séquentielles éprouve quant à elle des difficultés pour résoudre des problèmes comportant des ressources fortement cumulatives. En raison de ces deux écueils majeurs à la résolution de problèmes d'ordonnancement de projet généraux, nous proposons dans ce chapitre des modèles à même de pallier ces insuffisances.

L'objectif est donc ici de proposer des formulations alternatives permettant de résoudre des instances de projets comportant non seulement des ressources fortement cumulatives, mais avec aussi des horizons d'ordonnancement très grands. Dans ce chapitre nous présentons de nouvelles formulations du problème de gestion de projet à moyens limités (RCPSP) basées sur la programmation de linéaire en nombres entiers. Nous introduisons deux formulations de ce problème basées sur la notion d'événements déjà étudiée dans certains problèmes d'ateliers. De ces formulations inédites, nous déduisons d'autres variantes.

Ainsi, contrairement aux formulations utilisant des variables indicées par le temps (formulations DT et BDT par exemple), nous proposons ici deux nouvelles formulations qui utilisent des variables indicées par des événements. Celles-ci s'inspirent des travaux de Pinto et Grossmann [82] sur les problèmes de traitement par lots (*batch process problems*) et la formulation du problème de flow-shop proposée par Dauzère-Pérès et Lasserre [33]. Elles sont fondées également sur l'étude polyédrale sur l'ordonnancement de machine conduite par Lasserre et Queyranne [63].

Zapata *et al.* [99] proposent une formulation de type PLNE basée sur les événements pour la résolution du problème d'ordonnancement multi-projet multi-mode à ressource limitées (MRCMPSP). Ces formulations utilisent trois types de variables binaires : une première variable x^s_{pijn} qui permet de marquer le début d'exécution des activités ($x^s_{pijn} =$

1 si l'activité i du projet p débute à l'événement n avec la combinaison j), tandis que la deuxième variable x^{f}_{pijn} marque la fin de l'exécution. Le troisième type de variables, x^{p}_{pijn} indique les événements pendant lesquels l'activité est en cours d'exécution. Pour des tests dont les temps de résolution sont moins restreints que pour la mémoire, leur formulation obtient des résultats avantageux comparée aux autres types de formulations.

Dans ce chapitre, nous définissons la notion d'événement utilisée pour indicer nos variables. Puis nous présentons deux formulations inédites utilisant ces variables indexées par les événements que nous proposons pour la résolution du RCPSP. Nous étudierons quelques variantes de ces formulations. Après une étude analytique, nous finirons par une conclusion.

Le contenu de ce travail a fait l'objet des communications suivantes : [5, 57–59]

3.1 Notion d'événement

L'utilisation des événements pour indexer des variables a déjà fait l'objet de certaines études [33, 63, 82] dans le cadre des problèmes d'atelier avec des ressources disjonctives (machines). Son application inédite au RCPSP implique certaines subtilités propres à ce type de problème qui met en jeu des ressources cumulatives.

Si l'on considère qu'un événement se produit lorsqu'une activité débute ou termine son exécution, alors, à chaque activité, il est possible d'associer deux événements qui sont : un *événement de début d'exécution* et un *événement de fin d'exécution*. Ces deux événements caractéristiques des activités, nous permettent d'identifier l'exécution de chaque activité sur l'horizon d'ordonnancement du projet.

Cependant, dans un ordonnancement semi-actif qui consiste à ordonnancer les activités au plus tôt, avec des précédences de type début-fin sans temps de latence, la date de début (événement de début d'exécution) d'une activité correspond soit à l'instant zéro, soit coïncide avec la date de fin (événement de fin d'exécution) d'une autre activité. Par conséquent, on peut restreindre le nombre des événements au nombre des activités plus un, sachant que des études ont déjà démontré que l'ensemble des ordonnancements

semi-actifs est un ensemble dominant pour la recherche du makespan [90]. Ainsi, dans la suite de ce document, on désignera l'ensemble des événements par $\mathscr{E} = \{0, 1, \ldots, n\}$, ainsi $|\mathscr{E}| = n+1$ où n est le nombre d'activités.

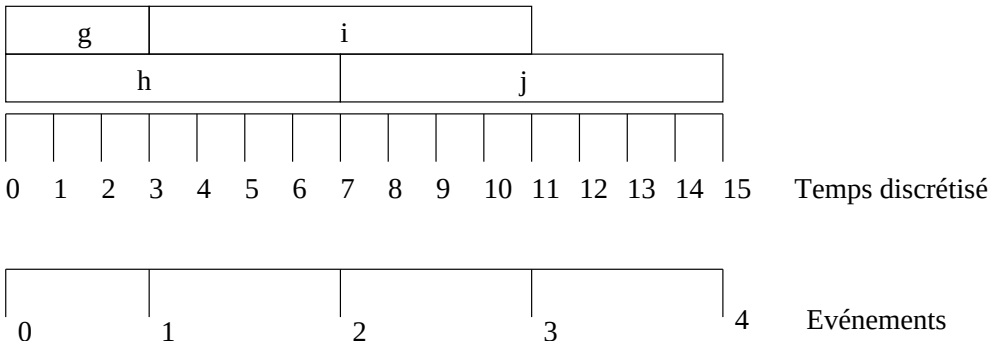

Figure 3.1 – Exemple d'ordonnancement à temps discret

Le diagramme de Gantt de la figure 3.1 représente un ordonnancement valide pour un problème de 4 activités, une ressource de capacité 2, et dont le $C_{\max}$ vaut 15. Cinq événements sont suffisants pour distinguer les phases caractéristiques de cet ordonnancement. Ceci met en évidence une diminution du nombre de variables, notamment des variables binaires, lorsqu'on passe des modèles utilisant des variables indexées par le temps, aux modèles comportant des variables indexées par des événements.

3.2 Formulation *Start/End* basée sur les événements (SEE)

À la différence des modèles à temps discret utilisant des variables binaires x_{it} stipulant si oui ou non l'activité i commence au temps t, notre formulation dite *Start/End* utilisant des variables indexées par les événements, se distingue par l'utilisation de deux types de variables binaires (x_{ie} et y_{ie}). Celles-ci déterminent l'exécution d'une activité dans le temps, à l'aide des événements. Plus précisément, une variable de binaire x_{ie} indique si oui ou non l'activité i débute son exécution à l'événement e, et une variable y_{ie} indique si oui ou non l'activité i termine son exécution à l'événement e.

Dans ce modèle, nous avons également besoin de deux types de variables continues t_e et b_{ek}. La variable t_e désigne la date de l'événement e. En se référant à l'exemple de

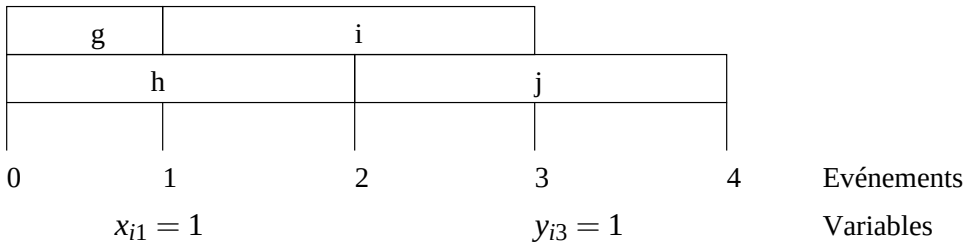

Figure 3.2 – Exemple de diagramme de Gantt utilisant des événements

la figure 3.1, on a : $t_0 = 0$, $t_1 = 3$, $t_2 = 7$, $t_3 = 11$ et $t_4 = 15$. Quant à la variable continue b_{ek}, elle indique le niveau de la ressource k utilisée entre les événements e et $e+1$.

Dans ce modèle, l'utilisation des activités fictives 0 et $n+1$ n'est pas nécessaire. Par conséquent, on ne considère que l'ensemble A des n activités (non-fictives), et donc $n+1$ événements.

3.2.1 Formulation

La formulation *Start/End* basée sur les événements (SEE), que nous proposons se présente de la manière suivante :

$$\min_{r,t,x,y} t_n \tag{3.1}$$

$$t_0 = 0 \tag{3.2}$$

$$t_f \geq t_e + p_i x_{ie} - p_i(1 - y_{if}) \qquad \forall (e,f) \in \mathscr{E}^2, f > e, \forall i \in A \tag{3.3}$$

$$t_{e+1} \geq t_e \qquad \forall e \in \mathscr{E}, e < n \tag{3.4}$$

$$\sum_{e \in \mathscr{E}} x_{ie} = 1 \qquad \forall i \in A \tag{3.5}$$

$$\sum_{e \in \mathscr{E}} y_{ie} = 1 \qquad \forall i \in A \tag{3.6}$$

$$\sum_{e'=e}^{n} y_{ie'} + \sum_{e'=0}^{e-1} x_{je'} \leq 1 \qquad \forall (i,j) \in E, \forall e \in \mathscr{E} \tag{3.7}$$

$$r_{0k} = \sum_{i \in A} b_{ik} x_{i0} \qquad \forall k \in R \tag{3.8}$$

$$r_{ek} = r_{(e-1)k} + \sum_{i \in A} b_{ik} x_{ie} - \sum_{i \in A} b_{ik} y_{ie} \qquad \forall e \in \mathscr{E}, e \geq 1, k \in R \tag{3.9}$$

$$r_{ek} \leq B_k \qquad \forall e \in \mathscr{E}, k \in R \tag{3.10}$$

$$t_e \geq 0 \qquad \forall e \in \mathscr{E} \tag{3.11}$$

$$r_{ek} \geq 0 \qquad \forall e \in \mathscr{E}, k \in R. \tag{3.12}$$

$$x_{ie} \in \{0,1\}, y_{ie} \in \{0,1\} \qquad \forall i \in A, \forall e \in \mathscr{E} \tag{3.13}$$

Il s'agit d'un programme linéaire mixte qui vise à minimiser la durée totale du projet (makespan), sous un ensemble de contraintes que nous décrivons ci-dessous.

3.2.2 Description des contraintes

La fonction-objectif est donnée par (3.1) et l'équation (3.2) fixe la date de l'événement 0 placé par convention à la date 0.

Proposition 3.2.1. *L'inéquation (3.3) assure que si l'activité i débute son exécution à*

l'événement e et la termine à l'événement f, alors le moment qui sépare l'événement f de l'événement e est au moins égal à la durée opératoire de l'activité i.

Démonstration. On peut montrer la relation suivante :
si $(x_{ie} = 1 \text{ et } y_{if} = 1)$ alors $t_f \geq t_e + p_i$.
On peut aussi montrer que :
lorsque la condition $(x_{ie} = 1 \text{ et } y_{if} = 1)$ alors n'est pas satisfaite, alors l'inéquation (3.3) n'élimine pas indûment de solutions acceptables. Si on introduit une variable $X_{ief} \in \{0,1,2\}$ telle que $X_{ief} = (x_{ie} + y_{if})$, on a :
$(x_{ie} = 1 \text{ et } y_{if} = 1) \Leftrightarrow X_{ief} = 2 \Leftrightarrow X_{ief} - 1 = 1$, avec $X_{ief} - 1 \in \{-1;0;1\}$. On peut réécrire la relation de départ de la manière suivante :
si $X_{ief} - 1 = 1$ alors $t_f \geq t_e + p_i$, Alors, on peut construire cette équation :
$t_f \geq t_e + p_i(X_{ief} - 1)$
qui sera équivalente à $t_f \geq t_e + p_i$
lorsque la condition ($X_{ief} - 1 = 1$) est respectée.
Par changement de variable, notre contrainte devient :
$t_f \geq t_e + p_i(x_{ie} + y_{if} - 1)$.
Pour les cas où la condition $(x_{ie} = 1 \text{ et } y_{if} = 1)$ n'est pas respectée, alors l'équation équivaut à :
$t_f \geq t_e - p_i$, quand $X_{ief} - 1 = -1$, c'est-à-dire quand $x_{ie} + y_{if} - 1 = -1$,
et à $t_f \geq t_e$, quand $X_{ief} - 1 = 0$, c'est à dire quand $x_{ie} + y_{if} - 1 = 0$.
La contrainte $t_f \geq t_e - p_i$ est alors dominée par la contrainte $t_f \geq t_e$ déjà incluse dans notre notre modèle (contrainte (3.4)), ce qui montre bien que l'inéquation (3.3) n'élimine pas de solutions acceptables pour notre problème. □

L'inéquation (3.4) impose que la date de l'événement *e* soit antérieure ou égale à la date de l'événement $e+1$. Ainsi, on ordonne les événements par convention.

Les contraintes (3.5) (respectivement (3.6)) imposent qu'une activité ne peut débuter (resp. ne peut se terminer) qu'une et une seule fois, c'est-à-dire n'est associée qu'à un et seul événement.

L'inéquation (3.7) représente les contraintes de précédence. Elle impose que :
si $\sum_{e'=0}^{e-1} x_{je'} = 1$ alors $\sum_{e'=e}^{n} y_{ie'} = 0 \quad \forall (i,j) \in E, \quad \forall e \in \mathscr{E}$.
En d'autres termes, pour toute paire d'activités (i,j) appartenant à l'ensemble de précédences E, l'événement de début d'exécution de l'activité j se produira en même temps ou après l'événement de fin d'exécution de l'activité i.

Les équations (3.8) et (3.9) représentent les contraintes de conservation des ressources. Les contraintes (3.9) (respectivement (3.8)) donnent la consommation totale r_{ek} (resp. r_{0k}) de la ressource k à l'événement e (resp. à l'événement 0). À l'événement 0, cette consommation totale est égale à la somme des consommations des activités qui démarrent à l'instant 0. Cependant, pour tout événement $e > 0$, elle est égale à celle de l'événement précédent ($r_{(e-1)k}$) augmentée de la somme des consommations des activités qui démarrent à l'événement e et diminuée de la somme des consommations des activités qui terminent leur exécution à ce même événement e.

Les contraintes (3.10) représentent les contraintes de ressources. Pour chaque événement e, elles limitent la consommation totale des ressources à la capacité totale disponible de cette ressource, et ce, pour chaque ressource k.

Cependant, notons que l'utilisation de variables continues r_{ek} n'est pas indispensable. On peut s'en passer dans la formulation du modèle. Pour ce faire, il suffit de les remplacer par leur expression dans les équations (3.8), (3.9), et dans l'inéquation (3.10). On substitue ainsi les contraintes de gestion des ressources (3.8), (3.9) et (3.10) par les deux inéquations suivantes :

$$B_k \geq \sum_{i \in A} b_{ik} x_{i0} \qquad \forall k \in K \qquad (3.14)$$

$$B_k \geq \sum_{e' \leq e} \sum_{i \in A} (b_{ik} x_{i,e'} - b_{ik} y_{i,e'}) \qquad \forall e \in \mathscr{E}, e \geq 1, k \in R \qquad (3.15)$$

3.2.3 Caractéristiques du modèle

Ce modèle Start/End basé sur les événements que nous proposons, semble prometteur du point de vue du nombre de variables binaires et de la simplicité de modélisation des contraintes de ressources. En effet, il comprend :

- $2n^2 + n$ variables binaires ;
- $n + 1$ variables continues ;
- $n(n+1)^2 + (n+1)(|E| + m) + 3n$ contraintes.

Il comporte donc un nombre polynomial de variables et de contraintes, et peut être classé parmi les modèles dits *compacts*.

Comme les modèles dits *séquentiels* [7, 35] (où une variable binaire x_{ij} décide de l'ordre relatif des activités *i* et *j*), le modèle Start/End est basé sur la logique qui consiste avant tout à trouver un ordre partiel ou global des activités, puis à appliquer une autre méthode de post-traitement pour affiner/déterminer la solution optimale. Un exemple d'une telle méthode de post-traitement est l'algorithme sériel [55] qui sélectionne les activités dans l'ordre de la liste (ordre global ou partiel) et ordonnance l'activité le plus tôt possible compte tenu des contraintes de précédence et de ressources, en tenant compte des activités ordonnancées aux étapes précédentes.

3.3 Formulation *On/Off* basée sur les événements (OOE)

Cette deuxième formulation basée sur la notion d'événements que nous proposons, utilise un seul type de variables binaires (z_{ie}) en plus des variables continues (t_e). La variable binaire z_{ie} permet d'identifier les intervalles pendant lesquels une activité débute ou est en cours d'exécution. Ainsi, la variable z_{ie} vaut 1 si l'activité *i* débute ou est en cours d'exécution immédiatement après l'événement *e*, sinon elle vaut 0. Quant à la variable t_e, elle détermine toujours la date de l'événement *e*.

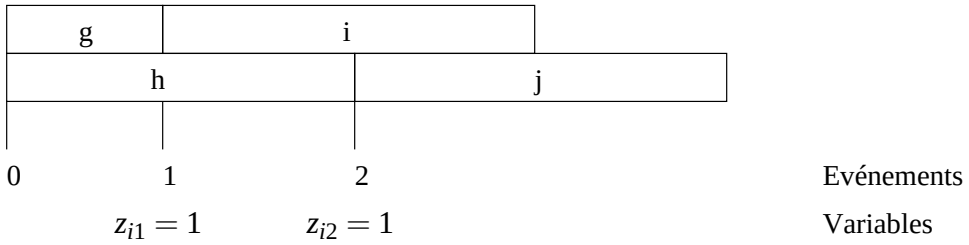

Figure 3.3 – Exemple de diagramme de Gantt de la formulation *On/Off*

Dans ce modèle, on prouve aisément qu'un ordonnancement optimal ne comportera

pas plus d'événements de début que d'activités existantes. De ce fait, on pourra restreindre l'ensemble des événements à n éléments $\mathscr{E} = \{0, 1, \ldots, n-1\}$, ainsi $|\mathscr{E}| = n$ sachant que l'utilisation des tâches fictives n'est pas indispensable. Le nombre d'événements est ici égal au nombre d'activités.

La figure 3.3 ci-dessus montre un exemple d'ordonnancement à quatre activités (g, h, i et j) et une seule ressource. On n'a besoin que de trois événements pour sa solution, car n'ayant pas d'activité qui démarre à la fin de l'exécution des activités i et j, il n'est donc pas nécessaire de leur associer des événements de fin.

3.3.1 Formulation

La deuxième proposition de formulation basée sur l'utilisation de variables indexées par des événements, dite formulation *On/Off*, se présente comme suit :

$$\min_{z,t,C_{\max}} C_{\max} \tag{3.16}$$

$$C_{\max} \geq t_e + (z_{ie} - z_{i(e-1)})p_i \qquad \forall e \in \mathscr{E}, \forall i \in A \tag{3.17}$$

$$t_0 = 0 \tag{3.18}$$

$$t_f \geq t_e + (z_{ie} - z_{i(e-1)} - (z_{if} - z_{i(f-1)}) - 1)p_i \quad \forall (e,f,i) \in \mathscr{E}^2 \times A, f > e \neq 0 \tag{3.19}$$

$$t_{e+1} \geq t_e \qquad \forall e \neq n-1 \in \mathscr{E} \tag{3.20}$$

$$\sum_{e'=0}^{e-1} z_{ie'} \leq e(1 - (z_{ie} - z_{i(e-1)})) \qquad \forall e \neq 0 \in \mathscr{E} \tag{3.21}$$

$$\sum_{e'=e}^{n-1} z_{ie'} \leq (n-e)(1 + (z_{ie} - z_{i(e-1)})) \qquad \forall e \neq 0 \in \mathscr{E} \tag{3.22}$$

$$\sum_{e \in \mathscr{E}} z_{ie} \geq 1 \qquad \forall i \in A \tag{3.23}$$

$$z_{ie} + \sum_{e'=0}^{e} z_{je'} \leq 1 + (1 - z_{ie})e \qquad \forall e \in \mathscr{E}, \forall (i,j) \in E \tag{3.24}$$

$$\sum_{i=0}^{n-1} b_{ik} z_{ie} \leq B_k \qquad \forall e \in \mathscr{E}, \forall k \in R \tag{3.25}$$

$$t_e \geq 0 \qquad \forall e \in \mathscr{E} \tag{3.26}$$

$$z_{ie} \in \{0,1\} \qquad \forall i \in A, \forall e \in \mathscr{E} \tag{3.27}$$

Il s'agit également d'un programme linéaire mixte qui vise à minimiser le makespan, tout en respectant les restrictions liées aux relations de précédences existant entre les activités et les restrictions liées à l'utilisation des ressources.

3.3.2 Description des contraintes

La définition de la fonction-objectif (makespan) de cette formulation est assurée par la contrainte (3.17). Elle impose que $C_{\max} \geq t_e + p_i$ si l'activité i est en cours d'exécution

juste après l'événement e, $\forall e \in \mathscr{E}, \forall i \in A$. C'est-à-dire qu'il faut que le $C_{\max}$ (makespan) soit supérieur ou égal à la date de fin de toute activité i en cours d'exécution juste après l'événement e.

Tout comme la contrainte (3.2) du modèle *Start/End*, la contrainte (3.18) fixe la date de l'événement 0 à l'instant 0.

Les contraintes (3.19) ont deux rôles essentiels. D'une part, elles permettent de lier les variables binaires z_{ie} aux variables continues t_e. D'autre part, elles assurent que si une activité i commence à l'événement e et se termine à l'événement f alors la date de l'événement f est au moins supérieure à la date de l'événement e augmentée de la durée opératoire de l'activité i. Ces contraintes sont obtenues en appliquant la même méthode qui a permis d'obtenir la contrainte (3.3).

La contrainte (3.20) tout comme la contrainte (3.4) impose que la date de tout événement e soit supérieure ou égale à la date de l'événement précédent $e-1$.

Les contraintes (3.21) et (3.22) représentent les contraintes de non-préemption des activités. Elles assurent la contiguïté des événements durant lesquels une activité reste en cours d'exécution. Ces contraintes permettent d'éliminer les solutions qui scinderaient l'exécution d'une tâche en plusieurs intervalles de temps non connexes.

La contrainte (3.21) est obtenue à partir de la relation suivante :

$$si \quad z_{i(e-1)} = 0 \quad et \quad z_{ie} = 1 \quad alors \quad \sum_{v=0}^{e-1} z_{iv} = 0, \quad \forall 0 \neq e \in E, \forall i \in A \tag{3.28}$$

On sait que $z_{i(e-1)} \in \{0,1\}$.

Par conséquent, on a $z_{ie} - z_{i(e-1)} \in \{-1,0,1\}$

et $1-(z_{ie}-z_{i(e-1)}) \in \{0,1,2\}$.

On sait également que :

$\sum_{v=0}^{e-1} z_{iv} \leq e$ sachant que z_{ie} est une variable binaire.

La relation 3.28 peut alors s'écrire de la manière suivante :

si $1-(z_{ie}-z_{i(e-1))} = 0$ alors $\sum_{v=0}^{e-1} z_{iv} = 0, \quad \forall 0 \neq e \in E, \forall i \in A$.

Ce qui nous permet de construire cette contrainte :

$\sum_{v=0}^{e-1} z_{iv} \leq e(1-(z_{ie}-z_{i(e-1)})), \quad \forall e > 0 \in E, \forall i \in A.$

De la même manière, la contrainte (3.22) est obtenue à partir de la relation suivante :
si $z_{i(e-1)} = 1$ et $z_{ie} = 0$ alors $\sum_{v=e}^{n-1} z_{iv} = 0, \quad \forall 0 \neq e \in E, \forall i \in A.$
Les contraintes (3.23) imposent que chaque activité soit exécutée au moins une fois pendant le projet. De cette manière, on s'assure que toutes les activités participent au projet, et qu'elles seront toutes ordonnancées.

Les contraintes (3.24) garantissent les relations de précédence entre les activités. Elles imposent que toute paire d'activité (i, j) appartenant à l'ensemble de précédences, si l'activité *i* débute ou est en cours d'exécution juste après l'événement *e*, alors l'activité *j* ne pourra pas débuter son exécution juste après l'événement *e*, ni avant.
Les contraintes de ressources (3.25) imposent que la somme des quantités de chaque ressource consommée par les activités en cours d'exécution à chaque événement *e* reste inférieure à la capacité disponible de la ressource.

3.3.3 Caractéristiques du modèle

Nous pouvons également classer ce modèle parmi les formulations compactes, car il comporte un nombre polynomial de variables et de contraintes, et même, beaucoup moins de variables que la proposition précédente (SEE). En effet, on dénombre

- n^2 variables binaires ;
- $(n+1)$ variables continues ;
- et $n^3 + n^2 + 3(n-1) + n(|E|+1)$ contraintes.

Cette caractéristique lui procure l'avantage de pouvoir traiter des projets ayant un horizon d'ordonnancement particulièrement étendu, ce qui n'est pas le cas pour les formulations à temps discret. Ces dernières, face à ce type de projet, génèrent en effet beaucoup de variables et contraintes. D'une part, le trop grand nombre de variables augmente la complexité dans l'arbre de recherche du branch-and-bound. Et d'autre part, le nombre important de contraintes peut provoquer des problèmes de mémoire.

A l'instar du modèle précédent, l'utilisation des activités fictives n'est pas utile pour ce modèle. Il est également basé sur la même logique de résolution qui consiste d'abord à trouver un ordre partiel ou global des activités, puis appliquer une méthode de post-traitement pour affiner/déterminer la solution optimale.

Une autre particularité de cette formulation est la facilité avec laquelle elle modélise les contraintes de ressource. En effet, il suffit de sommer les consommations des activités ayant la valeur de la variable binaire z_{ie} à 1 (i.e les activités en cours d'exécution), et d'imposer que cette somme soit inférieure à la capacité de la ressource concernée.

3.4 Etude analytique comparative des formulations basées sur les événements

Dans l'optique de mieux comprendre les modèles SEE et OOE, considérons l'instance illustrative du tableau 3.I ci-dessous, comprenant 10 activités et 2 ressources, dans lequel figurent les durées opératoires p_i, les consommations des ressources par les activités (lignes b_{i1} et b_{i2}). La capacité de chaque ressource vaut : $B_1 = B_2 = 3$. La figure 3.4

i	1	2	3	4	5	6	7	8	9	10
p_i	7	3	5	5	6	4	5	4	3	7
b_{i1}	0	2	3	3	2	1	1	1	1	3
b_{i2}	2	1	3	2	1	0	3	1	1	1
Successeurs	3	6,7	4,9	-	1	1	5,8	10	4	9

Tableau 3.I – Instance illustrative

montre le diagramme de Gantt d'un ordonnancement réalisable associé à cet exemple. Le makespan pour cette solution est égal à 38. On n'a besoin que de 9 événements pour SEE pour représenter cette solution ($n-1$ événements puisque les tâches 1 et 10 sont réalisées en parallèle) et juste 7 événements pour OOE (car elle ne tient pas compte des événements de fin d'activité).

Les tableaux 3.II, 3.III et 3.IV indiquent de manière synthétique l'instanciation des variables d'optimisation relatives aux activités 6 et 7, en fonction des formulations DT, DDT, SEE et OOE. Nous rappelons que la variable x_{it} est utilisée par les formulations DT et DDT (tableau 3.II), les variables x_{ie} et y_{ie} sont associées à SEE (tableau 3.III), et z_{ie} à OOE (tableau 3.IV).

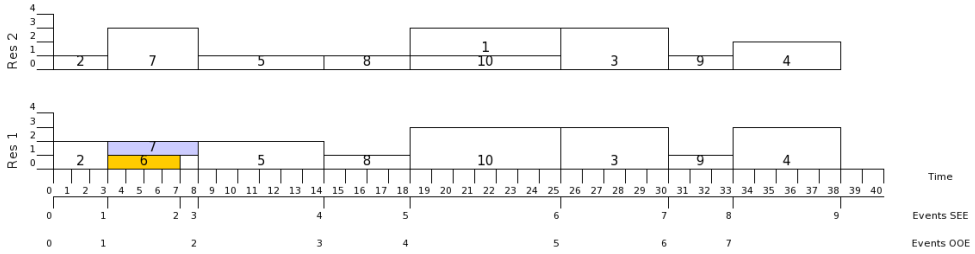

Figure 3.4 – Diagramme de Gantt d'une solution réalisable avec le temps discrétisé et les événements

t	2	3	4	5	6	7	8
x_{6t}	0	1	0	0	0	0	0
x_{7t}	0	1	0	0	0	0	0

Tableau 3.II – Les variables x_{it} des modèles DT et DDT

e	0	1	2	3
x_{6e}	0	1	0	0
y_{6e}	0	0	1	0
x_{7e}	0	1	0	0
y_{7e}	0	0	0	1

Tableau 3.III – Les variables x_{ie} et y_{ie} du modèle SEE

e	0	1	2
z_{6e}	0	1	0
z_{7e}	0	1	0

Tableau 3.IV – Les variables z_{ie} du modèle OOE

3.5 Variantes de formulations et “preprocessing”

Dans ce paragraphe, nous nous intéressons à un type de variante pour les formulations basées sur les événements que nous avons proposées ci-dessus. Nous présentons trois méthodes de *preprocessing* pour la formulation OOE.

3.5.1 Formulations à dates d’événements distinctes

Cette variante interdit que des événements soient simultanés. En d’autres termes, elle interdit que deux événements aient lieu à la même date (c’est-à-dire qu’on ne permet pas que $t_e = t_f$ avec $e \neq f$). Notons que cela n’interdit pas à deux activités de démarrer à une même date.

Il faut pour cela, en plus de certaines modifications supplémentaires, remplacer la contrainte (3.4) de la formulation SEE (respectivement la contrainte (3.20) de la formulation OOE) par la contrainte $t_{e+1} \geq t_e + 1, \forall e \in E$, pour avoir la formulation dite *SEE à dates d’événements distinctes* (resp. *OOE à dates d’événements distinctes*).

3.5.1.1 SEE à dates d’événements distinctes (SEE_DED)

La définition de la variante *SEE à dates d’événements distinctes* à partir de la formulation SEE implique certaines modifications supplémentaires. Il s’agit dans un premier temps d’ajouter au modèle une nouvelle variable continue $C_{\max} \geq 0$ associée au makespan. La fonction-objectif (t_n) est donc remplacée par la fonction-objectif

$$\min_{x,y,t,C_{\max}} C_{\max}. \tag{3.29}$$

Enfin il faut ajouter la contrainte supplémentaire

$$C_{\max} \geq t_e - H(1 - y_{ie}) \qquad \forall i \in A, \tag{3.30}$$

(3.31)

ainsi que la contrainte

$$C_{\max} \geq 0, \tag{3.32}$$

où H est l'horizon d'ordonnancement, ce qui exprime que le $C_{\max}$ doit être supérieur ou égal à la date de tout événement coïncidant à la fin d'exécution d'une activité.

3.5.1.2 OOE à dates d'événements distinctes (OOE_DED)

Pour obtenir la variante à dates d'événements distinctes de la formulation OOE, en plus du changement relatif à la contrainte (3.20), il faut aussi remplacer la contrainte (3.17) par cette nouvelle contrainte :

$$C_{\max} \geq t_e + H(z_{ie} - z_{i(e-1)} - 1) \qquad \forall i \in A \tag{3.33}$$

Il s'agit de la contrainte équivalente à la contrainte (3.30), qui impose que le makespan soit au moins égal à la date de fin de la dernière activité exécutée.

3.5.1.3 Avantages et inconvénients

Sur les instances cumulatives, où plusieurs activités peuvent s'exécuter en parallèle, si l'on parvient à éliminer par pré-traitement les événements non-utilisés, on diminue ainsi le nombre de variables et de contraintes.

Par contre, un inconvénient de ces variantes est lié à l'utilisation de l'horizon d'ordonnancement H (ou plutôt de la connaissance d'une borne supérieure pour le makespan) dans la formulation des contraintes. En effet, ceci a un effet dégradant sur la borne inférieure déterminée par la relaxation linéaire du modèle, car il est équivalente à l'utilisation d'un *big-M* dans les contraintes.

3.5.2 "Preprocessing" pour la formulation OOE

Le nombre de variables binaires des formulations basées sur les événements depend du nombre d'activités et du nombre d'événements pouvant être affectés à chaque ac-

tivité. Cependant, bien qu'ayant généralement moins de variables que les formulations indicées par le temps, ce nombre de variables peut toujours influencer les performances de la plupart des formulations de type PLNE. Pour cette raison, nous présentons trois méthodes de *preprocessing* pour la formulation OOE dans le but de fournir des directives sur la manière de réduire ce nombre de variables.

3.5.2.1 OOE_Prec

Cette formulation est obtenue en supprimant dans l'ensemble des événements possibles pour chaque activité, les événements pendant lesquels cette activité ne pourra pas être en exécution à cause de ses prédécesseurs. C'est-à-dire que un ou plusieurs de ses prédécesseurs sont en cours d'exécution ou n'ont toujours pas encore été exécutés. Symétriquement, on supprimera aussi pour cette activité, tous les événements pendant lesquels elle ne pourra pas être en cours d'exécution à cause de ses successeurs.

Soit $A(i)$ l'ensemble des prédécesseurs de l'activité i dans le graphe de précédence G, et $D(i)$ l'ensemble de ses successeurs.

Proposition 3.5.1. *Il existe une solution optimale de OOE telle que pour chaque activité* i *:* $\sum_{e=0}^{|A(i)|} z_{ie} = 0$ *et* $\sum_{e=n-|D(i)|+1}^{n} z_{ie} = 0$.

Démonstration. Lorsqu'on a n événements, il existe toujours une solution optimale telle que chaque activité débutera à un événement différent. C'est-à-dire que pour deux activités distinctes i et j, on aura :

$$z_{ie} - z_{i,e-1} = 1 \wedge z_{jf} - z_{j,f-1} = 1 \implies e \neq f.$$

Il est important de remarquer que cela n'interdit pas que $t_e = t_f$. De plus, si j est un prédécesseur de i dans G, alors l'événement affecté à j sera strictement antérieur à l'événement affecté à i : $t_e > t_f$. La propriété s'inspire de ces deux observations. □

Par conséquent, on obtient OOE_Prec en fixant donc ces variables suivantes à 0, et

on obtient ainsi une formulation correcte pour le RCPSP.

$$z_{ie} = 0, \quad i \in A, e \in \{0, \ldots, |A(i)|\} \cup \{n - |D(i)| + 1, \ldots, n\}. \tag{3.34}$$

3.5.2.2 OOE_Level

Cette formulation qui se présente comme une heuristique, introduit des variables réduisant l'espace de recherche. Typiquement, cette méthode de fixation de variables est fondée sur un ordre topologique des activités.

Soit v le nombre de niveaux dans le graphe de précédence G ; v est donc la longueur du plus long chemin en nombre d'arc dans le graphe G, entre 0 et $n+1$.

Soit $l^-(i)$ le *niveau au plus tôt* de l'activité i ; c'est-à-dire que l'activité i ne pourra en aucun cas appartenir à un niveau avant celui-ci. Il correspond au plus long chemin en nombre d'arcs entre 0 et i. Symétriquement, on note $l^+(i)$ le *niveau au plus tard* de i, c'est-à-dire que i ne pourra appartenir à aucun niveau se trouvant après ce niveau du graphe G, qui correspond aussi au plus long chemin en nombre d'arcs entre i et $n+1$.

Soient $N^-(i) = \{j \in A | l^+(j) < l^-(i)\}$ et $N^+(i) = \{j \in A | l^-(j) > l^+(i)\}$.

Bien que cela ne conduise pas à des solutions sous-optimales, intuitivement, on peut s'attendre à ce que toutes les activités d'un niveau de $l-1$ débutent avant que celles du niveau l ne débutent à leur tour. Ainsi, on peut définir la formulation OOE_Level en fixant à 0 les variables suivantes :

$$\sum_{e=0} z_{ie} = 0, \quad i \in A, e \in \{0, \ldots, |N^-(i)|\} \cup \{n - |N^+(i)| + 1, \ldots, n\}. \tag{3.35}$$

Notons que $A(i) \subseteq N^-(i)$ et $D(i) \subseteq N^+(i)$.

3.5.2.3 OOE_Reduced

Dans cette formulation, on réduit simplement le nombre d'événements en fixant le nombre maximum d'événements à $n\prime$, correspondant au nombre de dates de début distinctes d'une solution réalisable obtenue à l'aide d'une heuristique. Nous avons évidemment $n\prime < n$, mais nous ne savons pas s'il existe toujours une solution optimale impli-

quant au plus $n\prime$ événements. Il peut alors arriver que le nombre d'événements ne soit pas suffisant pour trouver une solution optimale. Il s'agit donc d'une formulation de type heuristique.

3.6 Conclusion

Dans ce chapitre, nous avons défini la notion d'événement utilisée auparavant pour indexer les variables dans la résolution de problèmes d'atelier.

Nous l'avons appliquée au RCPSP, à travers deux nouvelles formulations de type PLNE : la formulation Start/End basée sur les événements (SEE) et la formulation On/Off basée sur les événements (OOE).

Ces deux formulations présentent l'avantage d'être classées parmi les formulations de type compact, car elles comportent un nombre polynomial de variables et de contraintes ; en particulier, la formulation OOE utilise environ deux fois moins de variables que la formulation SEE. Elles n'utilisent pas les activités fictives usuelles et permettent également de modéliser facilement les contraintes de ressources. Dans le chapitre 5, consacré aux experimentations numériques , nous présenterons les résultats obtenus avec ces deux nouveaux modèles, et nous les comparerons avec ceux obtenus avec les modèles DT, DDT, et FCT.

CHAPITRE 4

LE RCPSP AVEC PRODUCTION ET CONSOMMATION DE RESSOURCES

Dans cette partie, nous nous intéressons à l'extension du RCPSP qui, au-delà des ressources renouvelables que nous avons considérées précédemment, comporte également des ressources pouvant être produites et consommées lors de l'exécution des activités. Nous désignons alors cette extension par le terme *RCPSP avec production et consommation de ressources*.

Il existe dans la littérature des travaux se rapportant à ce type de RCPSP avec production et consommation de ressources.

Le problème d'ordonnancement de projet avec contraintes d'inventaire de stock et contraintes temporelles généralisées, incluant à la fois des ressources renouvelables et non renouvelables, a été formalisé par Neumann et Schwindt [75] en 2002. Ils proposent également un branch-and-bound avec un filtrage par une heuristique de recherche en faisceau, pour la résolution de ce problème.

Carlier *et al.* [26] traitent ce type de problème (RCPSP avec production et consommation de ressources) incluant des contraintes de "time-lag" généralisées, en proposant un nouvel algorithme de liste.

Pour plus de détail sur l'état de l'art des travaux traitant de ce type de problème, nous renvoyons les lecteurs aux articles suivants [14, 19, 26, 61], dont celui de Laborie [61] définit les bases du concept du RTN (*"Resource Temporal Network"*), offrant un grande puissance de modélisation et qui a servi de support pour d'autres auteurs tel que Bouley *et al.* [19].

Dans le cadre de notre étude, dans un premier temps, nous définissons plus en détail cette extension de notre problème. Puis, nous présentons son intérêt et quelques-unes de ses applications dans le domaine industriel. Pour finir, nous exprimons les modifications que cette extension implique dans la formulation des différents modèles que nous avons

présentés dans les chapitres précédents : DT, DDT, FCT, SEE et OOE.

4.1 Description du problème et intérêt pratique

Dans le RCPSP classique, on a affaire généralement à des ressources renouvelables. Il s'agit de ressources qui redeviennent disponibles dans leur quantité originelle, une fois terminée l'exécution de l'activité les utilisant. Typiquement, une activité i consomme b_{ik} unités de la ressource k au début de son exécution, puis restitue exactement ces mêmes b_{ik} unités de k, une fois l'exécution de i terminée.

La particularité du RCPSP avec production et consommation de ressources, est qu'en plus d'utiliser les ressources décrites ci-dessus, on a aussi affaire à des ressources particulières. Il s'agit des ressources qui sont consommées (ou pas) au début de l'exécution d'une activité, dans une certaine quantité, puis reproduites dans une autre quantité à la fin de l'exécution de cette activité. Plus précisément, une activité i consomme c_{ip}^{-} unités de la ressource p au début de son exécution et en produit c_{ip}^{+} unités à la fin de son exécution. On désignera par P l'ensemble de ces ressources et C_p le niveau de stock initial de la ressource $p \in P$.

On peut distinguer plusieurs cas de figure concernant ces nouvelles ressources. Nous nous attardons particulièrement sur certains cas. Considérons d'abord un exemple avec une activité i de durée opératoire p_i, et deux ressources ($P = \{1,2\}$). L'activité i consomme c_{ip}^{-} unités de la ressource p et en produit c_{ip}^{+}, $\forall p = 1,2$.

- Si $c_{ip}^{-} = c_{ip}^{+}$, alors$p$ est une ressource renouvelable déjà prise en compte dans le RCPSP classique (voir figure 4.1, 1er cas).
- Dans le cas où on a $c_{ip}^{-} \neq c_{ip}^{+}$ (voir figure 4.1, 2e et 3e cas) :
  - Si $c_{ip}^{-} = 0$ et $c_{ip}^{+} > 0$, nous sommes face à un phénomène de production de resources pur et simple. Typiquement, dans la pratique, cette production est précédée de la consommation d'une certaine quantité d'une autre ressource. Exemple : $c_{i1}^{-} = 1$, $c_{i2}^{-} = 0$ et $c_{i1}^{+} = 0$, $c_{i2}^{+} = 10$. On consomme une unité de la resource 1 pour produire 10 unités de la ressource 2. Cette situation trouve des applications dans les industries de transformation de matières. Elle se ren-

contre également dans les usines de production d'énergie, aux deux détails près suivants :

1. D'une part, l'énergie est très souvent produite de façon continue au cours de l'exécution de l'activité productrice (voir figure 4.1, 4e cas).
2. D'autre part, en général, cette énergie peut être utilisée avant même l'achèvement de l'activité productrice.

– Si $c_{ip}^{-} < c_{ip}^{+}$, ce cas s'apparente au cas précédent, à la seule différence qu'on consomme au départ une certaine quantité du produit afin d'en produire davantage (voir figure 4.1, 3e cas).
– Si $c_{ip}^{-} > c_{ip}^{+}$, p est une ressource consommable (non renouvelable) telle qu'un budget, une matière première, etc. (voir figure 4.1, 2e cas).

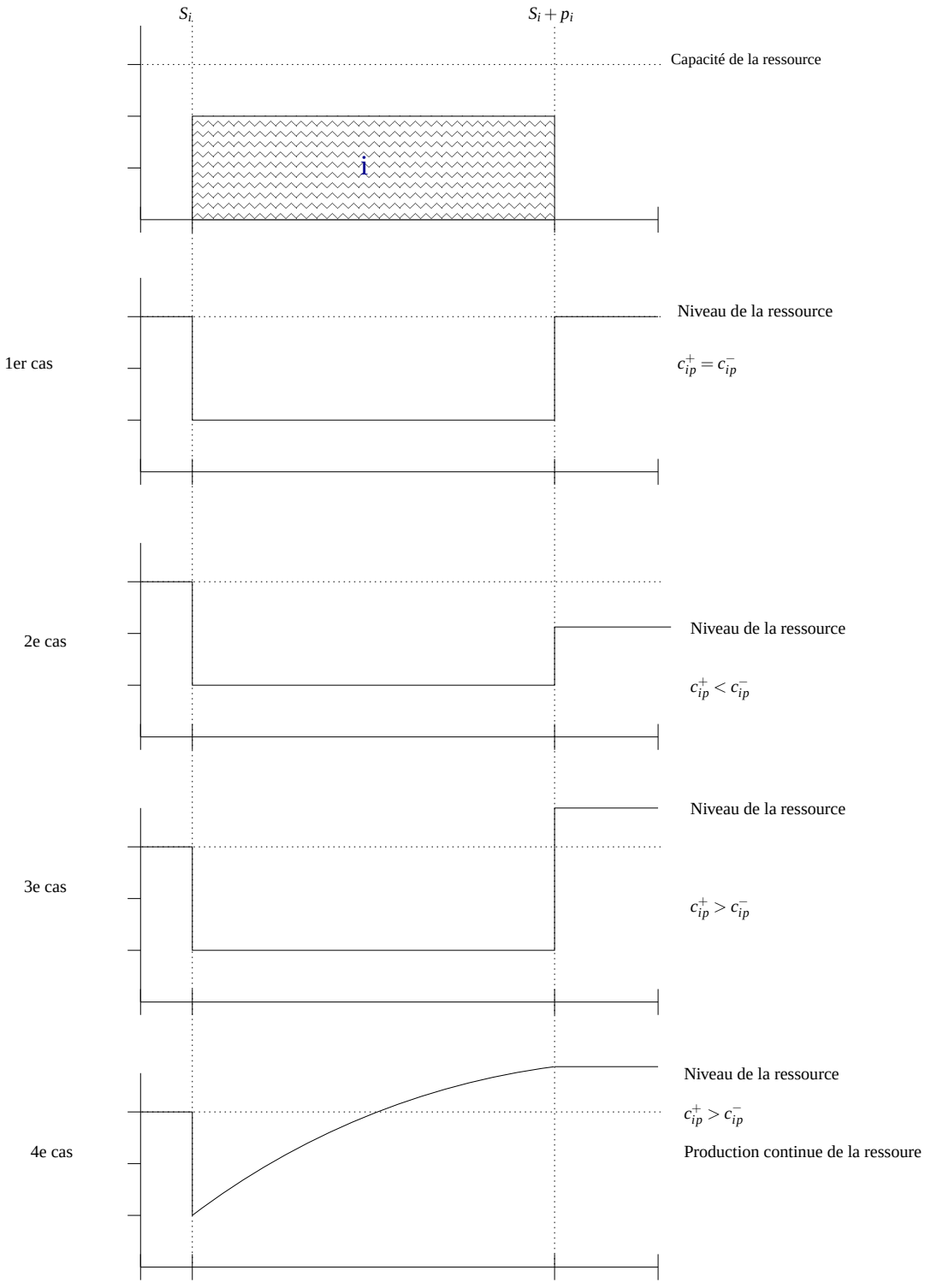

Figure 4.1 – Différentes formes de production de ressources

Même si l'étude de cette extension du RCPSP est motivée au départ par la volonté de résoudre des problèmes issus de *l'industrie des procédés* avec transformation de matière et gestion des bilans matières (nous y reviendrons dans la partie consacrée aux expérimentations), nous traitons le problème dans sa forme générale, avec des instances englobant tous les cas décrits ci-dessus.

4.2 Formulations à temps discret DT et DDT avec production et consommation de ressources

La prise en compte de l'aspect production de ressources implique l'ajout d'un certain nombre de modifications dans les modèles que nous avons vus précédemment. Cependant, compte tenu de la similarité des modèles DT et DDT dans la manière de gérer les ressources, les modifications ci-dessous concernent alors ces deux modèles. Ainsi, pour ces modèles DT et DDT, les modifications, en plus de l'ajout des variables c_{ip}^- et c_{ip}^+ (voir ci-dessus), consistent également en l'ajout d'une nouvelle variable s_{tp} et de trois nouvelles contraintes. Typiquement, la variable continue $s_{tp} \geq 0$ indique le niveau de stock de chaque ressource $p \in P$ à chaque instant t, et les contraintes additionnelles sont les suivantes :

$$s_{0p} = C_p - \sum_{i=1}^{n-1} x_{i0} c_{ip}^- \qquad \forall p \in P \qquad (4.1)$$

$$s_{tp} = s_{t-1,p} - \sum_{i=1}^{n-1} x_{it} c_{ip}^- + \sum_{i=1}^{n-1} x_{i,t-p_i} c_{ip}^+ \qquad \forall (t,p) \in H \times P, \quad t > 0 \qquad (4.2)$$

$$s_{tp} \geq 0 \qquad \forall (t,p) \in H \times P \qquad (4.3)$$

La contrainte (4.2) impose à chaque instant t, que le niveau de stock (s_{tp}) de chaque ressource $p \in P$ soit égal au niveau de stock à l'instant précédent ($t-1$) de cette ressource, augmenté de la somme des productions ($\sum_{i=1}^{n-1} x_{it} c_{ip}^-$) de cette même ressource par les activités finissant leur exécution à l'instant t, et diminué de la somme des consommations ($\sum_{i=1}^{n-1} x_{i,t-p_i} c_{ip}^+$) des activités débutant leur exécution à ce même instant t.

Quant à la contrainte (4.1), elle traite du cas particulier de la contrainte (4.2) à l'ins-

tant $t = 0$, c'est-à-dire quand on débute le projet.

En outre, une condition supplémentaire à respecter s'ajoutant au problème, consiste à imposer que les activités débutant leur exécution disposent de suffisamment de ressources pour être exécutées. Pour ce faire, on impose à travers la contrainte (4.3) que la variable s_{tp} reste toujours positive, exprimant ainsi le fait qu'on ne doit en aucun moment être en déficit de ressource.

On rappelle que T représente la valeur de l'horizon d'ordonnancement, n le nombre d'activités et l le nombre de ressources consommables.

De même que pour les variables de ressources du modèle à événements OOE du précédent chapitre, toutes les variables s_{tp} peuvent être substituées par leur valeur en fonction des x_{it}. Donc, on ne rajoute pas vraiment de variables. par contre, on ajoute réellement $(T+1)l$ contraintes supplémentaires, aux modèles DT et DDT pour prendre en compte l'aspect production et consommation de ressources, caractéristique de cette extension du problème.

4.3 Formulation basée sur les flots FCT avec production et consommation de ressources

À l'instar des modèles DT et DDT, la résolution de notre problème de RCPSP avec la production et la consommation de ressources nécessite la prise en compte de certaines modifications dans le modèle basé sur les flots (FCT). Ces modifications se matérialisent par l'ajout d'une variable continue et de cinq types de contraintes. Ainsi, on définit une variable continue M_{ijp} représentant la quantité de ressource $p \in P$ que l'activité i (à la fin de exécution), envoie à l'activité j (au début de son exécution). Comme l'exprime la contrainte (4.4) ci-dessous, cette quantité ne peut en aucun cas dépasser le minimum entre la quantité produite par l'activité i et la quantité consommée par l'activité j, si l'activité i précède l'activité j. Ainsi, conformément au principe de la formulation FCT, ces nouvelles ressources caractéristiques de cette extension de notre problème, sont également gérées par un système de recherche des flots dans un réseau. Les contraintes à

ajouter à la formulation sont les suivantes :

$$M_{ijp} \geq \min(c^+_{ip}, c^-_{jp}) x_{i,j} \qquad \forall (i,j) \in A^2, p \in P \qquad (4.4)$$

$$\sum_{j \in A \cup \{0\}} M_{ijp} = c^-_{jp} \qquad \forall i \in A \cup \{n+1\}, p \in P \qquad (4.5)$$

$$\sum_{i \in A \cup \{n+1\}} M_{ijp} = c^+_{ip} \qquad \forall i \in A \cup \{0\}, p \in P \qquad (4.6)$$

$$c^+_{0p} = C_k \qquad \forall p \in P \qquad (4.7)$$

$$c^+_{(n+1)p} = +\infty \qquad \forall p \in P \qquad (4.8)$$

La contrainte (4.5) impose que la somme des quantités de ressources $p \in P$ reçues des activités précédentes par une activité i, soit égale à la quantité de ressources qu'elle consomme pendant son exécution. Symétriquement, la contrainte (4.6) impose que la somme des quantités de ressources envoyées par l'activité i aux autres activités, soit égale à la quantité qu'elle produit. Cependant, à l'aide de la contrainte (4.7) (respectivement, contrainte (4.8)), pour chaque ressource $p \in P$, on fixe la production de cette ressource par l'activité 0 (respectivement l'activité $n+1$) à son niveau de stock initial (respectivement, à l'infini). En somme, ces modifications se traduisent par l'ajout d'aucune variable, mais de $(n^2+2n+2)l$ contraintes supplémentaires.

4.4 Formulations SEE avec production et consommation de ressources

Comme pour les formulations DT et DDT, pour la prise en compte de l'aspect production de ressource dans le RCPSP, on introduit de nouvelles variables continues $s_{ep} \geq 0$ indiquant le niveau de stock de chaque ressource $p \in P$ immédiatement après chaque événement e. Ainsi, d'une part, la modélisation analogue aux contraintes (4.1) et

(4.2) pour le modèle SEE est donnée par :

$$s_{0p} = C_p - \sum_{i=1}^{n-1} x_{i0} c_{ip}^{-} \qquad \forall p \in P \tag{4.9}$$

$$s_{ep} = s_{e-1,p} - \sum_{i=1}^{n-1} x_{ie} c_{ip}^{-} + \sum_{i=1}^{n-1} y_{i,e} c_{ip}^{+} \qquad \forall (e,p) \in \mathscr{E} \times P, \quad e > 0 \tag{4.10}$$

Ces contraintes permettent de calculer le niveau de stock de chaque ressource p à chaque événement e. Ce niveau de stock est égal à celui à l'événement précédent $e-1$, diminué des quantités consommées à l'événement e par les activités débutant leur exécution, et augmenté de la production de celles finissant leur exécution à l'événement e. Ainsi, l'utilisation des variables binaires x_{ie} indiquant le début des activités et les variables $y_{i,e}$ indiquant la fin d'exécution des activités, caractéristiques de ce modèle, simplifie l'expression de ces contraintes (4.9) et (4.10), dites *contraintes d'état des stocks*.

D'autre part, l'expression analogue à la contrainte (4.3) pour le modèle SEE avec production de ressources est la suivante :

$$s_{ep} \geq 0 \qquad \forall (e,p) \in \mathscr{E} \times P \tag{4.11}$$

Cette dernière permet d'assurer que les activités démarrant leur exécution disposent de suffisamment de la ressource pour être exécutées.

Ainsi, l'adaptation de ce modèle pour la résolution du RCPSP avec production et consommation de ressources nous impose l'ajout de $(n+1)l$ contraintes supplémentaires, mais d'aucune variable supplémentaire.

4.5 Formulations OOE avec production et consommation de ressources

Pour la formulation OOE que nous avons proposée, caractérisée par l'utilisation des variables binaires z_{ie} indiquant les événements pendant lesquels une activité i démarre ou reste en cours d'exécution, la prise en compte de l'aspect production de ressources nécessite non seulement l'ajout de nouvelles contraintes, mais aussi l'introduction de nouvelles variables continues. Nous définissons :

- s_{ep} la variable continue indiquant le niveau de stock de la ressource $p \in P$ à l'événement e
- p_{iep} la variable continue indiquant la quantité de ressources $p \in P$ *produite* par l'activité i à l'événement e
- u_{iep} la variable continue indiquant la quantité de ressources $p \in P$ consommée (*utilisée*) par l'activité i à l'événement e

Afin de simplifier l'expression de certaines contraintes, on utilisera temporairement des variables intermédiaires X_{ie} et Y_{ie}. Il ne s'agit en aucun cas de nouvelles variables à ajouter au modèle, mais plutôt des notations permettant de simplifier l'écriture des contraintes. On pourra facilement remplacer directement dans les différentes contraintes ces deux variables intermédiaires par leurs définitions :

$X_{i,e} := z_{ie} - z_{i,e-1}$;

$Y_{i,e} := z_{ie} - z_{i,e+1}$.

On notera que $X_{i,e} = -Y_{i,e-1}$, avec $X_{i,e} \in \{-1,0,1\}$ et $Y_{i,e} \in \{-1,0,1\}$. $X_{i,e} = 1$ (donc, $Y_{i,e} = -1$) signifie que l'activité i se termine à l'événement e. Parallèlement, $Y_{i,e} = 1$ (donc, $X_{i,e-1} = -1$) signifie que l'activité i finit son exécution à l'événement e. Dans les autres cas, on a $X_{i,e} = Y_{i,e} = 0$, signifiant que l'activité i ne débute, ni termine son exécution à l'événement e.

Pour le RCPSP avec production et consommation de ressources, les contraintes additionnelles à ajouter au modèle OOE, sont les suivantes :

$$p_{iep} \geq 0 \qquad \forall (e,i,p) \in \mathscr{E} \times A \times P \qquad (4.12)$$

$$p_{iep} \geq c^{+}_{ip} Y_{i,e-1} \qquad \forall (e,i,p) \in \mathscr{E} \times A \times P \qquad (4.13)$$

$$p_{iep} \leq c^{+}_{ip} z_{i,e-1} \qquad \forall (e,i,p) \in \mathscr{E} \times A \times P \qquad (4.14)$$

$$p_{iep} \leq c^{+}_{ip} (1 - z_{i,e}) \qquad \forall (e,i,p) \in \mathscr{E} \times A \times P \qquad (4.15)$$

$$u_{iep} \geq 0 \qquad \forall (e,i,p) \in \mathscr{E} \times A \times P \qquad (4.16)$$

$$u_{iep} \geq c^{-}_{ip} X_{ie} \qquad \forall (e,i,p) \in \mathscr{E} \times A \times P \qquad (4.17)$$

$$u_{iep} \leq c^{-}_{ip} z_{ie} \qquad \forall (e,i,p) \in \mathscr{E} \times A \times P \qquad (4.18)$$

$$u_{iep} \leq c^{-}_{ip} (1 - z_{i,e-1}) \qquad \forall (e,i,p) \in \mathscr{E} \times A \times P \qquad (4.19)$$

$$s_{ep} = s_{e-1,p} + \sum_{i \in A} p_{iep} - \sum_{i \in A} u_{iep} \qquad \forall (e,p) \in \mathscr{E} \times P, e > 0 \qquad (4.20)$$

$$s_{0p} = C_p + - \sum_{i \in A} u_{i0p} \qquad \forall p \in P \qquad (4.21)$$

$$s_{ep} \geq 0 \qquad \forall (e,p) \in \mathscr{E} \times P \qquad (4.22)$$

En somme, on utilise $2ln^2$ variables continues et $(6n^2 + n)l$ contraintes supplémentaires pour adapter le modèle OOE à la résolution du RCPSP avec production de ressources. Ces nombres importants de variables et de contraintes supplémentaires sont essentiellement dus au fait que ce modèle ne dispose pas de variable permettant l'identification nette et précise du début (respectivement de la fin) de l'exécution de chaque activité correspondant au point de consommation (resp. de production) de la ressource. Ainsi, pour déterminer la consommation (respectivement la production) des activités débutant (resp. terminant) leur exécution à l'événement e, on utilise à la fois les contraintes (4.13), (4.14) et (4.15) (resp. (4.17), (4.18) et (4.19)).

Explicitement, si une activité i termine son exécution à l'événement e, alors la contrainte (4.13) a pour vocation d'imposer que la valeur de la variable p_{iep} qui détermine sa production de ressource p à cet événement e, soit au moins égale à sa production de res-

source c_{ip}^{+}, tandis que les deux autres contraintes (4.14) et (4.15) imposent que la valeur de cette même variable, à ce même événement *e*, pour la même activité *i*, soit au plus égale à cette même production de ressource c_{ip}^{+}. Cela revient à imposer que la valeur de la variable p_{iep} soit égale à c_{ip}^{+}, si l'activité *i* termine son exécution à l'événement *e*, pour toute ressource $p \in P$. La combinaison de ces contraintes permet également d'annuler la valeur de cette variable à tout autre événement, afin de ne rendre significative que sa valeur à l'événement *e* considéré plus haut.

De même, dans le cas de la *consommation* de ressource, la contrainte (4.17) impose que la variable u_{iep} soit au moins égale à c_{ip}^{-} (la consommation de la ressource *p* de l'activité *i*), alors les contraintes (4.18) et (4.19) imposent qu'elle ne dépasse pas cette même valeur c_{ip}^{-}. Ici, on considère que l'activité *i* débute à l'événement *e* et se termine à un autre événement quelconque. Ainsi de façon analogue à la variable p_{iep}, il résulte de la combinaison des contraintes (4.16), (4.17), (4.18), et (4.19) que la variable u_{iep} sera égale à c_{ip}^{-} à l'événement *e*, et nulle pour tous les autres événements.

L'exemple de la figure ci-dessous permet d'observer l'effet cumulé de ces différentes contraintes sur le domaine des valeurs de la variable p_{iep}, tout en déterminant le point de production de ressource d'une activité *i*, qui commence son exécution à l'événement $e-4$ et la termine à l'événement *e*. Dans ce diagramme de Gantt partiel, l'activité *i* qui nous intéresse figure sous forme de briques inclinées.

Le tableau 4.I (ci-dessous) détaille les résultats affichés sur l'exemple de la figure 4.2 (ci-dessus), présente l'évolution des variables z_{ie} et Y_{ie} en fonction de l'événement e'. La

$e' =$, ou $e' \in$	$[0, e-5[$	$e-5$	$e-4$	$e-3$	$e-2$	$e-1$	e	$e+1$	$]e+1, n[$
$z_{ie'} =$	0	0	1	1	1	1	0	0	0
$Y_{ie'} =$	0	-1	0	0	0	1	0	0	0
$1 - z_{ie'} =$	1	1	0	0	0	0	1	1	1

Tableau 4.I – Comportement des variables

variable z_{ie} reste égale à 1 pour les événements $e-4, e-3, e-1, e$, car l'activité *i* débutant à l'événement $e-4$ et finissant à l'événement *e* reste en exécution juste après ces

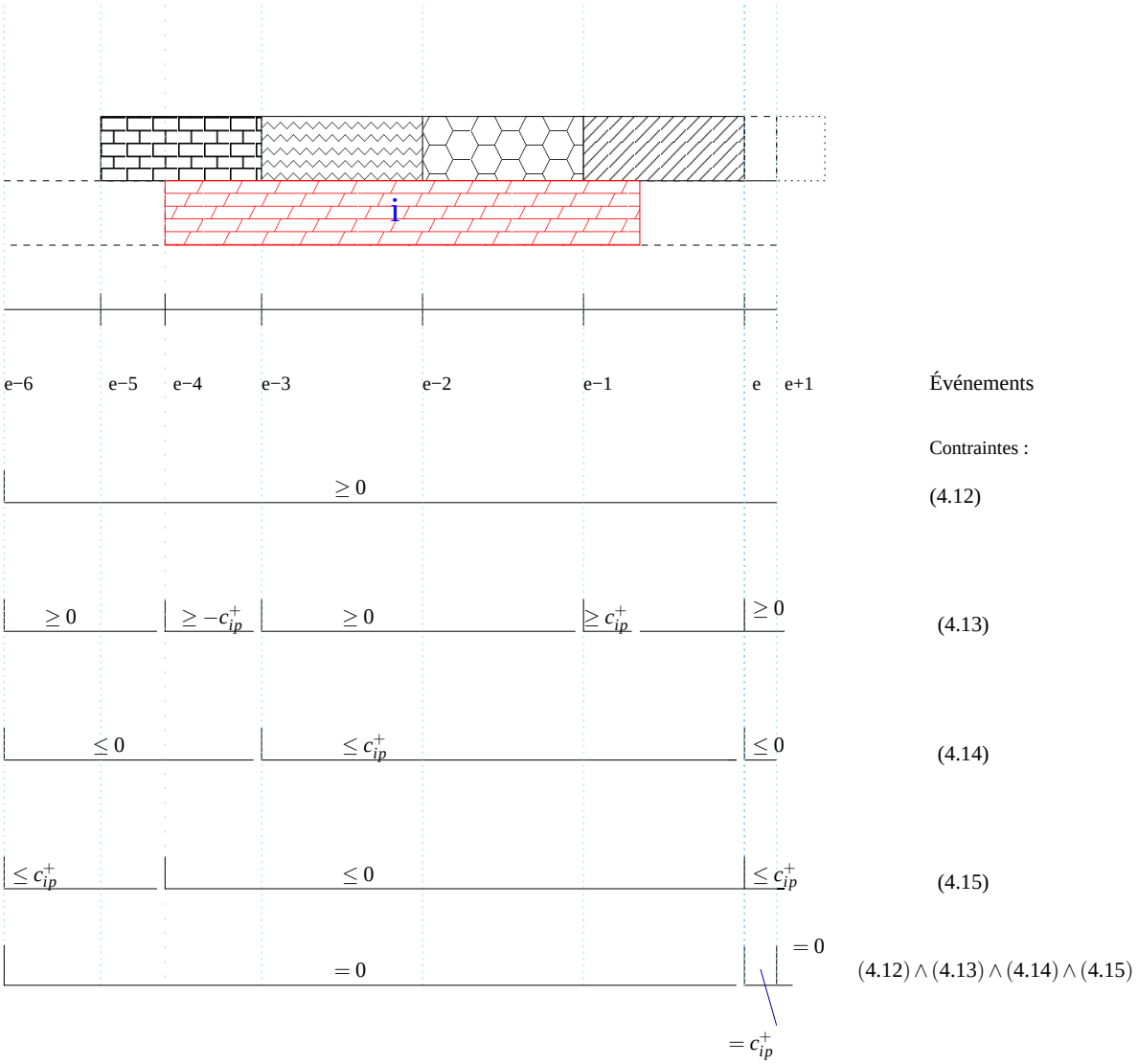

Figure 4.2 – Domaine des valeurs de la variable P_{iep}, pour une activité i exécutée entre événements $e-1$ et e

événements, mais ne l'est plus après l'événement e. De la valeur des z_{ie}, on peut déduire les valeurs de $Y_{ie'}$ et de $1-z_{ie'}$ entrant dans la formulation des différentes contraintes. Le tableau 4.II ci-dessous est la suite du tableau 4.I. Il permet, à partir de l'évolution des variables binaires z_{ie}, d'expliquer l'effet des contraintes (4.12), (4.13), (4.14) et (4.15) sur la détermination du point de production de ressource, à la fin de l'exécution de l'activité i à l'événement e, d'où les valeurs des variables P_{iep}.

Contrainte	$e' =$, ou $e' \in$	$[0, e-5[$	$e-5$	$e-4$	$e-3$	$e-2$	$e-1$	e	$e+1$	$]e+1, n[$
(4.12)	$P_{ie'p} \geq$	0	0	0	0	0	0	0	0	0
(4.13)	$P_{ie'p} \geq$	0	0	$-c_{ip}^{+}$	0	0	0	c_{ip}^{+}	0	0
(4.14)	$P_{ie'p} \leq$	0	0	0	c_{ip}^{+}	c_{ip}^{+}	c_{ip}^{+}	c_{ip}^{+}	0	0
(4.15)	$P_{ie'p} \leq$	c_{ip}^{+}	c_{ip}^{+}	0	0	0	0	c_{ip}^{+}	c_{ip}^{+}	c_{ip}^{+}
(4.12)$\wedge$ (4.13)$\wedge$ (4.14)$\wedge$ (4.15)	$P_{ie'p} =$	0	0	0	0	0	0	c_{ip}^{+}	0	0

Tableau 4.II – Interaction des contraintes

La contrainte (4.20) détermine le niveau de stock de la ressource $p \in P$ à l'événement e. Il est égal au niveau de stock à l'événement précédent, moins la somme des quantités produites de la même ressource par les activités en fin d'exécution, moins la somme de celles consommées par les activités en cours, tout ceci à l'événement e. Donc, il est également possible de supprimer les variables s_{ep} par leur valeur.
La contrainte (4.22) s'assure qu'une activité ne peut débuter son exécution que si elle dispose de suffisamment de ressource pour être exécutée. Pour finir, alors que la contrainte (4.12) (respectivement la contrainte (4.16)) interdit que les quantités produites (resp. consommées) soient négatives, la contrainte (4.22) permet de garantir de façon analogue aux contraintes (4.3) et (4.11), qu'on ne sera en aucun cas en situation de déficit de ressource.

4.6 Conclusion

Dans cette partie, nous avons pris en compte la possibilité de production de ressources lors de l'exécution des activités dans le RCPSP. Ainsi, après avoir donné une définition de cette extension de notre problème, nous avons proposé des modifications permettant d'adapter les formulations DT, DDT, FCT, SEE et OOE afin qu'elles puissent traiter cette extension, qui trouve diverses applications dans le domaine industriel.
Des résultats numériques évaluant les performances de ces formulations pour la résolution de diverses instances de ce problème, feront l'object d'un paragraphe dans le prochain chapitre (chapitre 5), portant sur les expérimentations.

CHAPITRE 5

EXPÉRIMENTATIONS

Dans ce chapitre entièrement consacré aux expérimentations numériques, nous décrivons tout d'abord, les indicateurs d'instances habituellement utilisés pour caractériser les instances que nous avons utilisées dans notre étude pour la comparaison des modèles proposés pour la résolution du RCPSP. Par la suite, nous présentons les instances que nous avons choisies pour nos tests. À partir de ces instances, nous proposons également deux nouveaux types d'instances ayant pour particularité des durées très longues. Les résultats numériques présentés par la suite, peuvent être regroupés en deux parties. La première partie concerne une étude comparative des formulations DT, DDT et FCT que nous avons présentées dans le chapitre 2, et les formulations SEE et OOE décrites dans le chapitre 3. La seconde partie des résultats concerne une étude sur l'extension du RCPSP au cas de production/consommation de ressources évoqué dans le chapitre 4. Tous les tests ont été effectués sur un PC DELL, XEON 5110 biprocesseur cadencé à 1,6 GHz, avec 4 GB de RAM, tournant sous le système d'exploitation FEDORA de Linux. Toutes les formulations ont été codées en C++, dans un environ ILOG-Concert Version 2.6. Nous avons utilisé le solveur commercial ILOG-Cplex version 11.1 pour les résolutions. Le temps limite de résolution pour chaque instance a été fixé à 500 secondes.

Une partie des résultats de ce travail a été consignée dans le chapitre de livre [6].

5.1 Le prétraitement des fenêtres de temps

Compte tenu que certaines formulations sont très sensibles à la longueur de l'horizon de l'ordonnancement, afin donc d'atténuer les effets de cette sensibilité, pour chaque formulation, nous associons à chaque activité i, une *date de début au plus tôt* (ES_i) et une *date de début au plus tard* (LS_i), calculées par prétraitement indépendamment de la résolution. La date de début au plus tôt ES_i d'une activité i correspond à la date avant

laquelle l'activité ne peut démarrer, tandis que la date de début au plus tard est la date à laquelle l'activité *i* doit avoir démarré. Ainsi, $[ES_i, LS_i]$ représente la *fenêtre de temps* de démarrage de l'activité . Notons que si chaque arc liant les paires d'activités $(i, j) \in E$ de l'ensemble de précédences est valué par la durée opératoire p_i de l'activité *i*, alors la fenêtre de temps valide pour chaque activité *i*, peut être calculée à l'aide d'un algorithme de recherche de chemins les plus longs dans un graphe, telle que : la date de début au plus tôt ES_i est égale à la longueur du plus long chemin du sommet initial 0 au sommet i ; de même, la date de début au plus tard LS_i est égale à la date de début au plus tôt ES_{n+1} du sommet terminal $n+1$, moins la valeur du plus long chemin entre le sommet i et le sommet terminal $n+1$.

Notons que la date de fin au plus tard de l'activité fictive de fin LS_{n+1} est calculée au moyen d'une heuristique. Nous avons choisi d'utiliser la méthode d'ordonnancement parallèle avec comme règle de priorité la plus petite date de fin au plus tard dans l'ordonnancement optimal sans contraintes de ressources (MINLFT) [55]

En plus de ces calculs basiques, nous utilisons d'autres techniques de réduction des fenêtres de temps, basées sur la propagation des contraintes de ressources. Plus précisément, nous lançons en prétraitement la partie propagation de contraintes du code de Demassey *et al* [35] qui implémente les techniques de sélection immédiate, *Edge Finding* sur les cliques de disjonction, et triplets symétriques. Nous renvoyons à Demassey *et al* [35] pour plus de références sur ces techniques.

Ainsi, par l'ajout de coupes ou de fixations de variables dans les différentes formulations, on impose que la date de début de chaque activité *i* soit prise dans sa fenêtre de temps. De plus, on réduit également la date de début au plus tôt de l'activité $n+1$ qui est une borne inférieure du makespan. On réduit ainsi le nombre de variables (principalement dans les formulations à temps discret), et on renforce les inégalités valides des formulations [35].

5.2 Les indicateurs d'instances et les instances

5.2.1 Les indicateurs d'instances

Apparus dans les années 60 pour caractériser les instances, les indicateurs d'instances pour le RCPSP peuvent être classés globalement selon quatre catégories décrites ci-dessous.

Indicateurs basés sur les précédences.

Il s'agit d'indicateurs orientés sur les relations de précédence. Ils décrivent essentiellement la densité et la complexité du graphe de précédence. Parmi ces indicateurs, on trouve :

– *Order Strength (OS)* [70]. Il définit la densité de la fermeture transitive (TE) du graphe de précédence. Il est formulé de la manière suivante :

$$\mathrm{OS} = |TE|/(n(n+1)/2).$$

Cet indicateur vaut 0 lorsqu'il n'y a aucune relation de précédence entre les activités. Dans ce cas, toutes les activités peuvent s'exécuter en parallèle. On parlera alors de *parallélisme total*.

En revanche, quand $OS = 1$ cela signifie que toutes les activités sont reliées entre elles par des relations de précédence. Dans ce cas, on dit que les activités sont *totalement ordonnées*.

Herroelen [46] a pu observer par des tests numériques, que le degré de difficulté décroît lorsque la valeur de OS croît. Intuitivement, on peut facilement imaginer que la taille de l'arbre de recherche décroît avec l'ajout de contraintes de précédence entre activités.

– *Network complexity (NC)*. Cet indicateur se formule ainsi :

$$\mathrm{NC} = \sum_{i \in A} |\Gamma_i \cup \Gamma_i^{-1}|/2n,$$

où la fonction Γ_i représente l'ensemble des successeurs de l'activité i et Γ_i^{-1} l'en-

semble de ses prédécesseurs. Cet indicateur correspond au nombre moyen d'arcs de précédence par activité. Autrement dit, il indique le nombre moyen de successeurs directs des activités. Cependant, même si Kolisch *et al.* [84] avancent que le degré de difficulté des instances décroît lorsque la valeur de l'indicateur OS croît, cette corrélation ne fait pas l'unanimité, car, face à d'autres résultats mitigés obtenus par d'autres chercheurs, cet indicateur n'a pas permis de tirer des conclusions [38, 86].

Indicateurs basés sur les ressources.

Ces indicateurs orientés sur les ressources, visent à définir les caractéristiques des ressources et leur influence sur le degré de difficulté des instances. Parmi ces indicateurs, on a :

– *Resource factor (RF)*. Il s'agit du nombre moyen normalisé de ressources par activité. Si l'on considère une variable binaire w_{ik} telle que $w_{ik} = 1$ si $b_{ik} > 0$ et $w_{ik} = 0$ sinon, alors on peut formuler cet indicateur comme :

$$\mathrm{RF} = \frac{\sum_{i \in \mathscr{A}} w_{ik}}{n|\mathscr{R}|}.$$

De ce fait, on peut en déduire que $0 \leq \mathrm{RF} \leq 1$. Kolisch *et al.* [84] montrent expérimentalement que la difficulté d'une instance croît lorsque RF croît. Cependant, Herroelen [45] remarque que la valeur de RF peut être biaisée s'il existe dans les instances un nombre significatif d'activités ne consommant aucune ressource.

Indicateurs basés sur le temps.

Cette autre catégorie, regroupant les indicateurs d'instances orientés sur le temps, sert à décrire l'impact du temps sur les caractéristiques des instances. Nous n'avons pas sélectionné d'indicateur purement de ce type mais des indicateurs associant plusieurs caractéristiques (voir ci-dessous)

Indicateurs hybrides.

Cette dernière catégorie d'indicateur fédère les indicateurs regroupant les caractéristiques d'au moins deux des catégories citées ci-dessus. Ils permettent d'avoir une idée

plus globale de la difficulté des instances. A titre d'exemple, on peut citer les indicateurs ci-dessous :

– *Resource Strength* [84]. Il combine les caractéristiques des indicateurs orientés sur les ressources (2e catégorie) et celles des indicateurs orientés sur le temps (3e catégorie). Défini pour chaque ressource $k \in R$, il se formule comme suit :

$$RS_k = (B_k - b_k^{\min})(b_k^{\max} - b_k^{\min})$$

où $b_k^{\min} = \max_{i \in A} b_{ik}$ est la plus grande consommation de la ressource k, et $b_k^{\max}$ le pic de demande de la ressource k dans l'ordonnancement au plus tôt basé sur les précédences :

$$b_k^{\max} = \max_{t \in \{0,\ldots,ES_{n+1}-1\}} \sum_{i \in A \cup \{0,n+1\}} b_{ik}.$$

Exprimant la disponibilité des ressources, $RS_k = 0$ pour une ressource k dont la capacité totale correspond à la capacité totale minimale de la ressource k garantissant la faisabilité du projet. Tandis que le cas $RS_k = 1$ correspond à l'absence totale de contrainte liée à la ressource k. C'est-à-dire, dans ce cas, que la consommation maximale de cette ressource k reste toujours inférieure à sa capacité totale disponible B_k.

Kolisch *et al.* [84] affirment que le temps moyen de résolution diminue continuellement avec RS, tandis que De Reyck et Herroelen [86] ont montré expérimentalement que la courbe de variation du temps CPU en fonction de RS a la forme d'une cloche (facile sur les bords et difficile au milieu, les instances où RS est proche de 0 étant beaucoup plus difficiles que celles où RS est proche de 1). Un inconvénient majeur de RS, est qu'il suffit qu'une activité ait une consommation maximale sur une ressource pour qu'on ait un RS = 0, indépendamment des autres données du problème.

– *Disjunction Ratio (DR)* [10]. Cet indicateur intègre les caractéristiques des indicateurs orientés sur les relations de précédence et celles des indicateurs orientés sur les ressources. Il a été défini par Baptiste et Le Pape pour distinguer les ins-

tances majoritairement cumulatives (DR faible) et les instances majoritairement disjonctives (DR élevé). Globalement, les auteurs constatent expérimentalement que les instances réputées difficiles de la littérature (voir ci-dessous) sont en fait fortement disjonctives (peu d'activités peuvent être exécutées en parallèle), ce qui peut produire un biais dans la résolution et un manque de représentativité des instances disponibles. Ils mettent ainsi en évidence l'intérêt de générer de nouvelles instances difficiles et fortement cumulatives (beaucoup d'activités pouvant être exécutées en parallèle). Cet indicateur est formulé comme suit :

$$\mathrm{DR} = |TE \cup D|/(n(n+1)/2)$$

avec TE la fermeture transitive du graphe de précédence, et D l'ensemble des paires d'activités en disjonction ($D = \{(i,j)|\exists k \in R, b_{ik} + b_{jk} > B_k\}$).

5.2.2 Les instances

L'évaluation des performances des modèles proposés dans le cadre de la résolution du RCPSP, nécessite l'utilisation de jeux de données (instances) communes. Aujourd'hui, les plus utilisées sont les instances proposées par Kolisch *et al.* [56], succédant à celles de Patterson devenues assez faciles à résoudre avec les nouvelles méthodes proposées. Cependant, divers types d'instances présentant des caractéristiques différentes existent et sont disponibles en ligne pour la plupart. Dans notre étude, nous avons utilisé les instances suivantes.

KSD. Il s'agit des instances proposées par Kolisch *et al.* [56] et ont été générées à l'aide de leur générateur de projet ProGen. Ce sont les instances les plus utilisées actuellement. Elles peuvent être classées en quatre grands groupes en fonction de leur nombre d'activités. Ainsi, il existe 480 instances pour chaque groupe de 30 activités (KSD30), de 60 activités (KSD60), de 90 activités (KSD90) et 600 instances de 120 activités (KSD120). Chaque instance comporte 4 ressources renouvelables, avec des durées opératoires choisies aléatoirement entre 1 et 10. Dans le cadre de notre étude, nous nous sommes intéressés uniquement aux 480 instances

de 30 activités (KSD30).

BL. Ces instances ont été proposées par Baptiste et Le Pape [10]. Typiquement, il s'agit d'un ensemble de 39 instances, parmi lesquelles 19 comportent 20 activités, et les 20 autres instances en comportent 25. Chaque instance comporte 3 ressources et chaque activité a une demande en ressource générée aléatoirement entre 0 et 60 de la capacité totale de chaque ressource. Les instances de 20 activités contiennent 15 contraintes de précédences ($|E| = 15$) générées aléatoirement aussi, alors que celles de 25 activités en contiennent 45 ($|E| = 45$). Conçues pour répondre aux critiques sur le caractère fortement disjonctif des instances KSD, les instances BL sont connues pour être fortement cumulatives.

PACK. Carlier et Néron [27] proposent un ensemble de 55 instances avec très peu de contraintes de précédences. Le nombre d'activités varie entre 17 à 35, pour 3 ressources. Les capacités des ressources vont de 5 à 10. Ces instances peuvent être regroupées en deux catégories. Dans la première, pour chaque ressource k, les consommations des activités sont générées aléatoirement entre 0 et B_k, ce qui induit un nombre significatif de disjonctions entre les activités. A contrario, dans la seconde catégorie, les consommations sont générées entre 0 et $B_k/2$, ce qui implique donc l'absence de disjonction et renforce le caractère cumulatif de ces instances. Les instances PACK caractérisées par un faible ratio de disjonction (DR) et un nombre faible de relations de précédence, sont connues comme des instances assez difficiles à résoudre [7].

5.2.3 Les instances modifiées

Les instances issues de la littérature n'ont pas toujours fait l'unanimité et ont même souvent été l'objet de certaines critiques. Par exemple, concernant les instances KSD, il apparaît que les instances les plus difficiles à résoudre sont celles ayant une valeur de l'indicateur RS assez faible, révélant alors de fortes disjonctions entre les activités, ce qui n'est pas représentatif des instances réelles du RCPSP.

Il est important aussi de souligner que la plupart des instances proposées ne comportent que des durées opératoires relativement courtes (comprises entre 0 et 10 pour les

KSD, par exemple). D'emblée, ceci confère un avantage aux formulations utilisant des variables indicées par le temps, car, dans ce cas, celles-ci ne génèrent pas beaucoup de variables et de contraintes, atténuant ainsi leur handicap lié au fait qu'en réalité, il s'agit de formulations classées parmi les modèles de type pseudo-polynomial (cf. chapitre 2). Alors que dans certaines réalités industrielles (pétrochimique ou pharmaceutique par exemple), on a parfois affaire à des durées assez longues pour certaines activités, avec de grandes disparités entre les durées. Ces cas se retrouvent également dans le domaine de l'industrie des procédés, et dans certains problèmes d'ordonnancement avec des traitements par lots [82].

Ces critiques ouvrent donc la voie à la proposition de nouvelles instances. Ainsi, dans un souci de plus grande représentativité des différents cas, nous proposons deux nouveaux types d'instances, qui ne sont en réalité que des versions modifiées de certaines instances existantes : PACK_d et KSD15_d, Nous donnons ci-dessous une description plus précise de la méthode utilisée pour générer ces nouvelles instances.

Pour générer une instance (B) à partir d'une instance existante (A), considérons les paramètres x, y, b et a suivants tel que :

1. On sélectionne les x premières activités non-fictives de l'instance A (on laisse de côté les autres activités ainsi que les contraintes de précédence qui leur sont adjacentes).
2. Pour les activités sélectionnées, on connecte les activités sans prédécesseurs à l'activité 0 et les activités sans successeurs à l'activité $x+1$.
3. Puis, on sélectionne de façon aléatoire y activités parmi les x activités non-fictives et on multiplie leur durée par un coefficient $a+b$, où b est un nombre aléatoire généré entre 0 et 1 et a un *facteur multiplicatif de la durée* (défini ci-dessous pour chaque type d'instance). Il s'agit d'une valeur arbitraire permettant d'augmenter considérablement les durées.

Les durées obtenues sont arrondies au nombre entier le plus proche.

KSD15_d. Ces instances ont été créées à partir des KSD30 selon la procédure évo-

quée ci-dessus, avec les paramètres suivants : $x = 15$, ce qui correspond à la moitié des activités initiales du KSD30 ; $y = 7$ c'est-à-dire que parmi les 15 activités sélectionnées, seule la durée opératoire de 7 activités d'entre elles sera augmentée, avec la valeur du facteur multiplicatif fixé à $a = 25$.

On obtient ainsi 480 instances de taille plus réduite ($n = 15$), mais avec des durées opératoires plus grandes, allant ainsi de 1 à approximativement 250 unités de temps.

PACK_d. Ce deuxième type d'instances proposées a été obtenu par la modification des instances PACK, avec les paramètres suivants : Nous avons utilisé la totalité des activités initiales $x = n$, avec n le nombre d'activités dans les instances PACK. On modifie la durée opératoire de 10 de ces activités ($y = 10$), en utilisant un facteur multiplicatif $a = 50$. On obtient donc 55 instances de même taille que les PACK originaux, mais avec des durées d'activités atteignant parfois le millier.

Toutes ces nouvelles instances sont disponibles à l'adresse internet suivante [1].

5.2.4 Caractéristiques des instances

Le tableau 5.I ci-dessous résume les caractéristiques des différentes instances utilisées en fonction des indicateurs d'instances évoqués précédemment.

Tableau 5.I – Valeurs moyennes des indicateurs pour chaque type d'instance utilisée

	KSD30	BL	PACK	KSD15_d	PACK_d
$\|V\|$	32	22 - 27	17 - 35	17	17 - 35
$\|R\|$	4	3	2 - 5	4	2 - 5
T	34 - 130	14 - 34	23 - 139	187 - 999	644 - 3694
OS	0.34 - 0.69	0.25 - 0.45	0.13 - 0.48	0.34 - 0.64	0.13 - 0.48
NC	1.5 - 2.13	1.45 - 2	1.5 - 1.72	1.18 - 1.82	1.50 - 1.72
RF	0.25 - 1	0.5 - 0.77	1	0.25 - 1	1
RS	0.14 - 1	0.16 - 0.55	0.08 - 0.53	0.18 - 1	0.08 - 0.48
DR	0.36 - 0.9	0.25 - 0.45	0.19 - 0.94	0.35 - 0.9	0.19 - 0.94
PR	10	5	19	250	1138

Notons en outre que :

PR est le rapport de la plus grande durée sur la plus petite durée. Il représente l'amplitude des écarts entre les durées opératoires des activités : $PR = \max_i p_i / \min_i p_i$;
$|V|$ représente le nombre d'activités dans les instances ;
$|R|$ est le nombre de ressources dans les instances ;
T est la longueur de l'horizon d'ordonnancement. Plus précisément, T est une borne calculée comme étant la date de début au plus tard, LS_{n+1}, de la tâche fictive de fin du projet (donnée par l'heuristique parallèle avec la règle MINLFT).

Il ressort de ce tableau que :

- Les instances KSD30 comprennent 32 activités (dont 2 fictives), 4 ressources, avec des horizons d'ordonnancement comprises entre 34 et 130 unités de temps. Le rapport PR entre la durée maximale et minimale est de 10. Elles ont en moyenne plus de relations de précédence ($1.5 \geq NC \geq 2.13$ et $0.34 \geq OS \geq 0.69$), et sont également plus disjonctives ($0.36 \geq DR \geq 0.90$) que les autres instances.
- Les instances BL, quant à elles, comportent de 22 à 27 activités, 3 ressources, avec des horizons très courts compris entre 14 et 34 unités de temps. Le rapport entre la durée maximale et la durée minimale vaut 5 ; donc, les durées sont sensiblement égales. Il s'agit d'instances fortement cumulatives, avec un peu moins de relations de précédence, comparées aux KSD30.
- Les instances PACK (17 à 35 activités, 2 à 5 ressources, horizons de temps entre 23 et 139) sont également cumulatives. Elles comportent très peu de relations de précédence. Le rapport entre la durée max et la durée min est plus grand, comparé aux KSD30, et toutes les ressources sont utilisées au moins une fois par toutes les activités ($RF = 1$).
- Les instances KSD15_d de 17 activités, 4 ressources, ont des horizons de temps compris entre 187 et 999 unités de temps, avec un rapport de 250 entre les durées max et min. Le graphe de précédence est moins dense que pour les KSD30, mais les instances sont tout aussi disjonctives.
- Les instances PACK_d gardent quasiment les mêmes caractéristiques que les instances PACK à une différence majeure près, car les PACK_d ont des horizons de

temps beaucoup plus grands, allant de 644 à 3694 (au lieu de 23 à 139 pour les PACK).

5.3 Résultats numériques

Les résultats que nous présentons dans ce paragraphe proviennent de deux types d'expériences que nous avons menées. Les premières concernent les formulations PLNE proposées dans les chapitres 2 et 3 (DT, DDT, FCT, SEE et OOE), pour la résolution du RCPSP basique (c'est-à-dire sans consommation et production de ressources), tandis que le second type d'expérience concerne ces mêmes formulations, avec les mêmes instances, mais pour la résolution du RCPSP avec consommation et production de ressources, évoqué dans le chapitre 4.

5.3.1 RCPSP sans production/consommation de ressources

Dans cette première partie des tests, nous comparons les formulations DT, DDT, FCT, SEE et OOE sur les instances KSD30, BL, PACK, KSD15_d et PACK_d, selon deux types de résolution. Dans un premier temps, nous effectuons des tests pour une résolution en relaxation continue et, dans le second temps, nous considérons une résolution exacte.

5.3.1.1 Résultats - relaxation continue

Dans cette première série de tests, concernant la résolution en relaxation continue (relaxation des contraintes d'intégralité donnant des bornes inférieures du problème), nous obtenons les résultats suivants présentés dans le tableau ci-dessous :

Résolues : Le pourcentage d'instances dont une valeur de la relaxation est trouvée au bout de 500 secondes.

Δ**CPM** : Le pourcentage de l'écart moyen entre la valeur du chemin critique (borne inférieure) et la valeur de la relaxation trouvée. Cet indicateur constitue une référence absolue permettant de comparer différents modèles.

Tableau 5.II – Résultats de la relaxation continue

Instances	**Modèles**	**Résolues**	**ΔCPM**	ΔES_{n+1}	**Temps (s)**
KSD30	DDT	100	8.9	0.31	0.78
	DT	100	7.99	0.04	0.04
	OOE	100	7.94	0.00	0.39
	SEE	100	7.94	0.00	3.01
	FCT	97	7.96	0.00	6.94
PACK	DDT	100	148.88	17.99	1.47
	DT	100	142.23	12.57	0.18
	OOE	100	128.94	0.00	0.22
	SEE	100	128.94	0.00	1.32
	FCT	100	128.94	0.00	3.53
BL	DDT	100	13.09	13.09	0.12
	DT	100	5.53	5.53	0.05
	OOE	100	0.00	0.00	0.17
	SEE	100	0.00	0.00	1.09
	FCT	97	0.00	0.00	2.25
KSD15_d	OOE	100	7.69	0.00	0.04
	SEE	100	7.69	0.00	0.05
	FCT	100	7.69	0.00	0.10
	DT	71	6.78	0.00	0.28
	DDT	15	21.22	22.05	0.52
PACK_d	OOE	100	80.55	0.00	0.15
	SEE	100	80.55	0.00	0.41
	FCT	100	80.55	0.00	3.79
	DT	0	-	-	-
	DDT	0	-	-	-

ΔES_{n+1} : Le pourcentage de l'écart moyen entre la valeur de la date de début au plus tôt de l'activité fictive de fin de projet (borne inférieure calculée en prétraitement) et la valeur de la relaxation trouvée.

Temps (s) : Le temps moyen requis pour trouver une valeur de la relaxation, en secondes.

Les modèles sont classés en termes de performance selon les critères suivants par ordre d'importance : le plus grand nombre d'instances résolues (plus grand Résolues), puis la plus grande variation par rapport à la valeur du chemin critique (plus grand ΔCPM) et pour finir, le meilleur temps de résolution (plus petit Temps (s)).

Les résultats présentés dans le tableau 5.II montrent que les formulations à temps discret fournissent les meilleurs résultats sur les instances classiques telles que les KSD30, les BL et les PACK, qui sont des instances à faibles horizons d'ordonnancement. Ainsi, les meilleures bornes inférieures calculées sont celles fournies par la formulation DDT, suivie de DT. Les formulations basées sur les événements et celle basée sur les flots produisent de très faibles bornes, et n'arrivent pas à améliorer les valeurs des bornes inférieures induites par le prétraitement par les fenêtres de temps. Ces résultats confirment la dominance des formulations à temps discret sur les instances classiques en conformité avec les précédentes études effectuées par d'autres chercheurs [35, 94].

Cependant, sur les instances non-classiques telles que les KSD15_d, la performance de la formulation DDT chute à 15 , tandis que celle de la formulation DT chute à près de 70 . Elles sont pires sur les PACK_d, car elles n'arrivent pas à les résoudre du fait du trop grand nombre de variables et de contraintes générées.

5.3.1.2 Résultats - résolution exacte

La seconde série de tests porte sur la résolution exacte, qui consiste à calculer la solution optimale pour chaque instance. Les résultats obtenus concernent non seulement les formulations présentées auparavant (DT, DDT, FCT, SEE et OOE), mais aussi les résultats obtenus à partir de la méthode spécifique exacte de Laborie [60] basée sur la recherche d'ensembles critiques minimaux (MCS). Les résultats sont présentés dans le tableau 5.III ci-dessus, comportant les abréviations suivantes :

Entières : Pourcentage de solutions entières (pas prouvées optimales) trouvées au bout de 500 secondes de résolution.

Opt : Pourcentage de solutions optimales trouvées en moins de 500 secondes de résolution.

Écart : Pourcentage de l'écart moyen entre la solution trouvée et la solution optimale déjà prouvée (ou à la meilleure solution connue).

Δ**CPM** : Pourcentage de l'écart moyen entre la solution trouvée et la valeur du chemin critique.

Temps Opt (s) temps moyen mis pour trouver une solution optimale.

Les formulations sont rangées par ordre de performance selon le critère : le plus grand nombre d'instances donnant lieu à une solution entière (plus grande valeur de Entières), puis le plus grand nombre de solutions optimales trouvées (Opt), le plus petit écart par rapport à la solution optimale prouvée (Écart), le plus petit écart à la valeur du chemin critique (ΔCPM) et pour finir le plus petit temps mis pour le résolution (Temps Opt (s)). Cependant, la méthode basée sur MCS étant particulière (méthode incrémentale qui fournit à chaque itération une borne inférieure du makespan, que l'on augmente au fur et à mesure que l'on considère des contraintes supplémentaires alors que les autres méthodes fournissent une solution qu'elles essaient d'améliorer en diminuant la valeur du makespan), elle ne renvoie qu'une borne inférieure si elle ne trouve pas de solution au bout des 500 secondes. La méthode MCS sera donc traitée différemment. Ainsi, les résultats de la méthode MCS ne tiennent compte que des solutions optimales trouvées (Opt) et non pas du nombre de solutions entières trouvées (Entières). De ce fait, pour chaque type d'instances on affichera en gras la meilleure valeur Opt obtenue.

Au vu des résultats affichés dans le tableau 5.III, deux tendances se dégagent :

– Pour les instances classiques KSD30, PACK et BL (instances avec des durées opératoires courtes), les meilleures formulations de type PLNE (dans l'ordre, meilleure valeur de Entières, Opt et Temps Opt) sont respectivement la formulation DDT, suivie de la formulation DT. Toutefois, on remarque que sur les instances KSD30 et PACK, le modèle MCS (non-classé parmi les modèles de type PLNE) affiche

Tableau 5.III – Résultats de la résolution exacte (branch and bound)

Instances	Formulations	Entières	Opt	Écart	ΔCPM	Temps Opt (s)
KSD30	DDT	91	82	0.47	8.91	10.45
	DT	86	78	0.55	6.74	12.76
	FCT	67	62	0.16	3.76	22.66
	OOE_Prec	46	30	1.69	13.65	52.31
	OOE	33	24	1.22	7.00	112.62
	SEE	3.1	2.9	0.24	0.61	123.62
	MCS	-	**97**	0.00	11.48	7.39
PACK	DDT	95	**76**	1.08	199.02	63.39
	DT	85	55	0.49	203.58	48.24
	OOE_Prec	55	5	3.25	227.19	18.92
	OOE	49	9	2.89	231.29	61.78
	FCT	2	0	1.28	14.49	-
	SEE	0	0	-	-	-
	MCS	-	25	0.00	149.81	115.88
BL	DDT	100	100	0.00	32.40	13.68
	DT	100	100	0.00	32.40	37.93
	OOE_Prec	54	0	7.26	40.30	-
	OOE	49	0	7.90	41.65	-
	FCT	21	3	6.14	30.64	310.58
	SEE	8	0	12.81	29.96	-
	MCS	-	**100**	0.00	32.40	3.29
KSD15_d	OOE_Prec	99.8	86	0.00	10.02	6.49
	FCT	99	94	0.02	9.02	12.06
	OOE	99	83	0.01	10.14	4.68
	SEE	92	76	0.15	9.86	13.04
	DT	55	54	0.23	4.31	12.10
	DDT	1	1	0.00	2.63	3.34
	MCS	-	**100**	0.00	10.18	0.07
PACK_d	OOE	60	18	1.26	120.13	75.58
	OOE_Prec	60	14	1.62	117.56	54.35
	FCT	7	7	0.00	0.00	60.88
	SEE	4	4	0.00	0.00	215.08
	DT	0	0		-	-
	DDT	0	0		-	-
	MCS	-	**38**	0.00	50.59	72.34

la meilleure performance. En outre, sur les instances PACK qui sont des instances très difficiles à résoudre, on remarque que les formulations indicées par le temps présentent de biens meilleures performances que le modèle MCS. En particulier, la formulation DDT résout à l'optimum trois fois plus d'instances que le modèle MCS. Ces résultats inattendus soulignent le progrès sans cesse croissant des solveurs PLNE et du coup confortent l'intérêt de continuer à creuser la piste des formulations de type PLNE pour la résolution du RCPSP.

La formulation à temps continu basée sur les flots, FCT, présente la meilleure performance parmi les formulations à temps continu uniquement sur les instances KSD30, mais pas sur les autres (PACK et BL). La formulation OOE, et sa variante avec prétraitement OOE_Prec, renvoient une solution entière pour environ la moitié des instances, soit bien mieux que FCT mais beaucoup moins bien que DDT et DT. Il n'est pas inutile de rappeler que le prétraitement appliquée à OOE_Prec a beaucoup plus d'effet sur les instances comportant beaucoup plus de relations précédences, telles que les instances KSD30. La formulation SEE présente de très mauvais résultats sur cette catégorie d'instances.

– Pour les nouvelles instances comportant de longues durées opératoires (KSD15_d et PACK_d), les performances des formulations à temps discret (DT et DDT) chutent. Elles n'arrivent qu'à résoudre très peu d'instances, et l'ordre de performance entre elles change également. Ainsi, la formulation DT génère moins de contraintes de précédence et devient meilleure que DDT qui en génère davantage. Quant au modèle MCS, il fournit le plus grand nombre de solutions optimales :100 des KSD15_d et 38 des PACK_d.

  Si on se focalise sur les formulations de type PLNE, même si FCT affiche les meilleures scores en termes de solutions optimales pour les KDS15_d, OOE_Prec (accessoirement OOE) fournit le plus grand nombre de solutions entières et de meilleures qualités (plus petites valeurs d'écart avec la meilleure solution connue). Elles nécessitent globalement moins de temps pour la résolution. Par exemple sur les KSD15_d, elles sont deux fois plus rapides.

  Pour les instances PACK_d (particulièrement difficiles), on remarque que le mo-

dèle MCS résout à l'optimum 30 des instances ce qui est la meilleure performance pour les solutions optimales. Les performances de la formulation FCT s'écroulent sur ces instances. Cependant, si on tient compte de tous les critères à la fois, on se rend compte que OOE et OOE_Prec trouvent des solutions entières pour 60 des instances, avec un faible écart par rapport à la meilleure solution connue, sur un temps de résolution assez court. Cela leur confère un avantage par rapport à tous les autres modèles (MCS compris). On note également qu'elles résolvent la quasi-totalité des instances avec une performance proche de celle du modèle MCS (qui a la meilleure performance). On retrouve ici le bénéfice de notre formulation OOE qui comporte deux fois moins de variables binaires que celle proposée par Zapata *et al.* dans [99].

On constate également que les instances PACK_d sont plus faciles à résoudre que les instances PACK (cf. résultats de MCS et des modèles basés sur les événements), montrant que l'introduction de durées hétéroclites (certaines très courtes et d'autres très longues) ne rend pas forcément les instances plus difficiles.

Nous résumons dans le tableau 5.IV les résultats obtenus. Ce tableau définit un ordre de préférence ('X $\succ$ Y' signifie que la formulation X est meilleure que la formulation Y) entre les formulations du type PLNE que nous avons testées pour une résolution exacte sur les instances KSD30, PACK, BL, PACK_d et KSD15_d. Par souci de lisibilité, on désignera par OOE_x soit la formulation OOE, soit sa variante avec prétraitement OOE_Prec. Une étude comparative dédiée aux différentes variantes de la formulation OOE sera l'objet du paragraphe suivant.

Tableau 5.IV – Synthèse des résultats des expériences

	Fort DR	Faible DR
Faible PR	DDT $\succ$ DT $\succ$ FCT $\succ$ OOE_x $\succ$ SEE	DDT $\succ$ DT $\succ$ OOE_x $\succ$ FCT $\succ$ SEE
Fort PR	FCT,OOE_x $\succ$ SEE $\succ$ DT $\succ$ DDT	OOE_x $\succ$ FCT $\succ$ SEE $\succ$ DT $\succ$ DDT

Il ressort des résultats présentés ci-dessus que, pour résoudre de manière exacte par la PLNE des instances avec de faibles durées opératoires (faible PR), les formulations à temps discret, notamment la formulation DDT, semblent être les plus indiquées. A

contrario, pour des instances avec un fort ratio de disjonction (fort DR), comportant de longues durées opératoires, la formulation à temps continue basée sur les flots, ou les formulations basées sur les événements semblent être les mieux indiquées. Cependant, si ces instances comportant des durées opératoires très longues sont plutôt fortement cumulatives, alors la formulation OOE conviendrait le mieux.

5.3.1.3 Étude comparative des variantes de OOE (résolution exacte)

Dans le tableau 5.V, les résultats d'une étude comparative des différentes variantes de OOE est proposée. Le critère choisi est le nombre et la qualité des solutions entières obtenues au bout de 500 secondes. L'utilisation de simples techniques heuristiques fondées sur des prétraitements (OOE_Level and OOE_Reduced) permettent d'augmenter le nombre de solutions entières trouvées, sans détériorer la qualité de l'écart. Au contraire, pour quelques instances, elle l'améliore. On observe ainsi que OOE_Reduced obtient les meilleurs résultats sur les instances BL, alors que OOE_Level s'avère plus efficace sur les instances KSD30 et PACK. Ceci témoigne du fait qu'il serait intéressant de mener des recherches plus poussées dans l'optique de trouver des heuristiques plus sophistiquées sur la base de la formulation OOE que nous avons proposée.

5.3.2 RCPSP avec production/consommation de ressources

Nous présentons maintenant les résultats des tests effectués pour évaluer les extensions à l'aspect de consommation et de production des ressources des formulations de type PLNE (DT, DDT, FCT, SEE et OOE) présentées dans le chapitre 4.

Ne disposant pas de jeux de données (instances) correspondant exactement au type de problème que nous étudions, nous avons utilisé également les mêmes instances KSD30, PACK, BL, KSD15_d et PACK_d avec quelques modifications.

Ces modifications consistent à générer pour chaque activité de chaque instance, trois nouvelles ressources (consommées et produites). Chacune de ces nouvelles ressources (non-renouvelables) est caractérisée par sa quantité c_{ip}^{-} consommée (respectivement sa quantité c_{ip}^{+} produite) au début (respectivement à la fin) de l'exécution de chaque activité,

Tableau 5.V – Résultats de la résolution exacte des variantes de OOE

Instances	**Formulations**	**Entières**	**Gap**	**ΔCPM**
KSD30	OOE_Level	52	1.68	14.72
	OOE_Reduced	48	1.31	10.08
	OOE_Prec	46	1.69	13.65
	OOE	33	1.22	7.00
PACK	OOE_Level	55	4.57	206.66
	OOE_Prec	55	3.25	227.19
	OOE_Reduced	53	4.19	212.62
	OOE	49	2.89	231.29
BL	OOE_Reduced	77	3.84	36.68
	OOE_Level	72	7.08	41.32
	OOE_Prec	54	7.26	40.30
	OOE	49	7.90	41.65
KSD15_d	OOE_Prec	99.8	0.00	10.02
	OOE_Level	99.6	0.02	10.05
	OOE_Reduced	99.4	0.01	10.05
	OOE	98.6	0.01	10.14
PACK_d	OOE_Level	69	1.44	123.38
	OOE_Prec	60	1.62	117.56
	OOE	60	1.26	120.13
	OOE_Reduced	54	1.93	122.97

générée aléatoirement entre 0 et 10. À chacune de ces ressources est également associé un stock initial (capacité initiale) C_p. Ces différentes caractéristiques de ces nouvelles ressources sont générées aléatoirement sous ces deux contraintes essentielles :

- aucune quantité consommée ne doit être supérieure au stock initial. Ceci autorise n'importe quelle activité à démarrer le projet ;
- afin de ne pas rendre le problème plus disjonctif qu'il ne l'était avant, on s'assure que la somme des quantités produites reste supérieure à la somme des quantités consommées.

5.3.2.1 Résultats

Les résultats obtenus à l'issue des tests numériques sont consignés dans le tableau 5.VI.

L'analyse de ces résultats montre que :

- Pour les instances KSD30, la formulation à temps discret présente les meilleurs résultats. DDT affiche la meilleure performance. Elle trouve des solutions entières pour 84 des instances, avec 63 de solutions optimales. Moins efficace que DDT et DT, la formulation à temps continu basée sur les flots produit d'assez bonnes performances. En revanche, les performances des formulations à événements sont très faibles sur ce type d'instance. Par l'observation de ces résultats qui sont globalement en baisse par rapport à ceux obtenus avec le RCPSP sans consommation et production de ressource, on constate bien que l'ajout de ces nouvelles ressources rend ces instances plus difficiles à résoudre.
- Concernant les instances PACK, on constate que le modèle OOE trouve le plus grand nombre de solutions entières, battant même les formulations à temps discret DT (deuxième meilleure performance) et DDT (troisième meilleure performance). Ces résultats représentent un changement notoire par rapport aux résultats du RCPSP sans consommation et production de ressource. Cependant, si on réduit le critère de performance au seul nombre de solutions optimales (Opt), DDT suivie de DT fournissent les meilleurs scores.
- Les résultats sur les instances BL confirment la même tendance qu'avec le RCPSP

Tableau 5.VI – Résultats de la résolution exacte du RCPSP avec consommation et production de ressource

Instances	Formulations	Entières	Opt	Écart	ΔCPM	Temps Opt (s)
KSD30	DT	84	63	9.62	25.20	52.60
	DDT	82	71	0.13	7.65	83.87
	FCT	69	20	44.66	70.35	289.39
	SEE	2	0	179.86	179.86	-
	OOE	1	0	65.96	65.96	-
PACK	OOE	94.55	1.82	18.59	277.83	110.85
	DT	90.91	18.18	43.45	365.49	126.63
	DDT	47.27	32.73	1.33	245.66	168.04
	SEE	29.09	0	3.16	134.13	-
	FCT	9.09	0	4.82	96.41	-
BL	DDT	94.87	48.72	0.49	47.62	125.55
	DT	87.18	38.46	48.88	119.82	108.60
	OOE	74.36	0	25.63	87.56	-
	SEE	41.03	0	31.11	88.51	-
	FCT	20.51	0	25.58	72.51	-
KSD15_d	FCT	100	93.94	0.12	10.15	18.26
	OOE	100	79.80	0.10	10.14	62.33
	SEE	98.99	75.76	0.32	10.30	49.34
	DT	0	0	-	-	-
	DDT	0	0	-	-	-
PACK_d	OOE	94.55	7.27	0.74	171	212.34
	SEE	23.64	5.45	1.35	63.50	278.53
	FCT	12.73	3.64	5.48	32.47	143.35
	DT	0	0	-	-	-
	DDT	0	0	-	-	-

sans consommation et production de ressource, donc une domination des formulations à temps discret DT et DDT. En revanche, contrairement à DDT et à DT qui chutent en passant du RCPSP basique au RCPSP avec consommation et production de ressource (nombre de solutions entières de 100 à 94 pour DDT et de 100 à 87 pour DT), on remarque une progression notable pour les modèles à événements (nombre de solutions entières de 40 à 74.36 pour OOE et de 8 à 41.03 pour SEE).

– Sur les instances KSD15_d, la formulation FCT est la meilleure, suivie de près par OOE. Cependant, l'ajout de nouvelles ressources sur des instances comportant déjà de très longues durées opératoires rend les modèles à temps discret (DT et DDT) très exigeants en mémoire et donc incapables des traiter ces instances dans les conditions de tests évoquées plus haut (dans notre estimation, il faudrait sans doute une machine comportant au moins 7 Go de RAM pour être à même de les résoudre).
– Identiquement au RCPSP basique, pour les instances PACK_d, les formulations OOE puis SEE et FCT fournissent les meilleures performances. En outre, sur ces instances, SEE devient meilleure que FCT.

De façon globale, on constate que les performances des modèles à événements sont plus élevées ici que dans les RCPSP basique, à l'exception des instances KSD30, alors que les performances des autres types de formulations diminuent.

La formulation OOE présente quasiment la meilleure performance sur deux types d'instances (PACK et KSD15_d), la deuxième meilleure performance sur un type d'instance (KSD15_d), la troisième sur un autre type (BL). Ceci nous permet de conclure que globalement, elle est la mieux adaptée pour ce type de problème.

5.4 Conclusion

Dans ce chapitre, à partir de cinq types d'instances, nous avons mené quatre études comparatives portant sur cinq formulations de type PLNE, dont deux inédites basées sur des événements que nous proposons dans le chapitre 3 : la formulation *Start/End Event-based* (SEE) et la formulation *On/Off Event-based* (OOE). Les trois autres formulations

provenant de la littérature, sont présentées dans le chapitre 2. Deux d'entre elles utilisent les variables binaires indicées par le temps (la formulation basique à temps discret DT et la formulation désagrégée à temps discret DDT) et la troisième est la formulation à temps continu basée sur les flots FCT.

Les problèmes de type RCPSP impliquant de longues durées opératoires sont courants dans l'industrie mais ne sont pas représentés dans les instances classiques disponibles qui servent de référence pour tester les modèles proposés. Aussi, nous avons proposé dans cette étude deux nouveaux types d'instances comportant de telles caractéristiques : KSD15_d et PACK_d.

Les résultats de notre première étude, portant sur la résolution de la relaxation continue de ces cinq formulations, sur les cinq types d'instances choisies (KSD30, PACK, BL, KSD15_d et PACK_d) montrent que les formulations à temps discret (notamment la formulation DDT) obtiennent de bons résultats sur les instances classiques. Par contre, sur les instances de longues durées, les programmes linéaires deviennent impraticables. Les formulations à temps continu ont comme prévu une relaxation très faible et l'introduction d'inégalités valides dans ces formulations pour obtenir des bornes inférieures acceptables semble une voie prometteuse.

La seconde étude, portant sur la résolution exacte de ces mêmes instances, confirme la tendance dégagée lors de l'étude précédente sur les instances classiques. Cependant, on remarque d'assez bonnes performances des formulations à événements, même si elles restent inférieures à celles de DT et de DDT. De plus, sur les instances à longues durées, les formulations à événements restent toujours les meilleures. Comparée au modèle MCS proposé par Laborie (l'une des meilleures méthodes spécifiques), en termes de solutions entières trouvées, OOE notamment, se rapproche des performances de cette dernière.

La troisième étude comparant les performances des quelques variantes très simples de la formulation OOE, permet de conclure que des travaux de recherche dédiés à l'amélioration de celle-ci conduiraient sûrement à accroître davantage ces performances.

La dernière étude porte sur l'évaluation de ces mêmes formulations sur le problème du RCPSP avec consommation/production des ressources. Les résultats montrent que, sur les instances à longues durées, OOE est la formulation la plus indiquée pour cette

extension du RCPSP par rapport aux autres modèles que nous avons testés. De plus ses performances par rapport au RCPSP classique sont globalement moins dégradées que celles des autres formulations sur la plupart des instances.

CHAPITRE 6

LE PROBLÈME DE TRANSBORDEMENT

La mondialisation, la délocalisation des sites de productions et le développement des moyens de communication et de transport favorisent la complexité des réseaux d'approvisionnement et de distribution dans les échanges commerciaux. Par conséquent, les biens et les personnes parcourent de plus en plus de distance avant d'atteindre leur destination finale. De par sa flexibilité, l'utilisation des infrastructures routières pour le transport national (continental dans certains cas) des marchandises occupe une place importante dans ces échanges. Cependant, entre l'approvisionnement chez les fournisseurs et la livraison aux destinataires finaux, il existe parfois un certain nombre d'étapes à franchir, telles que les opérations liées aux *problèmes de transbordement*.

Le problème de transbordement (*"cross-docking"* en anglais) est un problème logistique qui correspond à l'ensemble des opérations permettant l'échange de produits ou de passagers entre deux ou plusieurs moyens de transport transitant par des centres intermédiaires appelés *centres de transbordement*.

Un des enjeux les plus importants pour les entreprises impliquées dans ces échanges, est la maîtrise des coûts et des délais tout au long de la chaîne logistique. Pour ce faire, nombre d'entre elles expriment le besoin permanent de disposer de techniques, de méthodes et d'outils d'optimisation (logiciels d'aide à la décision) de plus en plus performants, visant alors à réduire les délais et les coûts de fonctionnement, dans un environnement de plus en plus concurrentiel. Tels sont donc les objectifs principaux des problèmes de transbordement, que nous abordons dans ce dernier chapitre de thèse.

Dans ce chapitre, nous nous intéressons au problème de transbordement dans sa version comprenant une seule porte d'entrée et une seule porte de sortie. À cet effet, nous présenterons le problème, ses applications industrielles, quelques travaux de l'état de l'art

et pour finir nous exposerons une nouvelle méthode de type branch-and-bound pour sa résolution.

Ce travail a fait l'objet d'une communication [69] au 13th Symposium on Information Control Problems in Manufacturing (INCOM'09), Russia.

6.1 Définitions et généralités

Les principales fonctions exécutées dans un entrepôt sont la réception des produits, leur stockage, la préparation des commandes et leur expédition. Ce faisant, certaines d'entre elles, telles le stockage et la préparation, sont des étapes plus sensibles et donc plus susceptibles d'entraîner une augmentation des coûts dans toute la chaîne logistique.

6.1.1 Définition

Le *crossdock* ou *entrepôt de transbordement* est défini comme une plate-forme de transbordement qui reçoit des produits en provenance de plusieurs fournisseurs (voir figure 6.1). Ces produits y sont déchargés, triés, regroupés en fonction de leur destination de livraison et envoyés. Ces types d'entrepôt comportent généralement trois grandes parties :

- Les *quais (ou les portes) d'arrivée*, là où accostent les camions contenant les produits en provenance des fournisseurs. Il s'agit donc des endroits où sont déchargés les produits.
- La *zone de stockage ou d'entreposage*. Les produits en attente d'envoi y sont triés, classés et stockés momentanément.
- *les quais (ou les portes) de sortie*. On y charge dans les camions les produits prêts à être envoyés chez les clients.

La durée moyenne de séjour d'un produit dans un entrepôt de transbordement est relativement courte (24 heures par exemple).

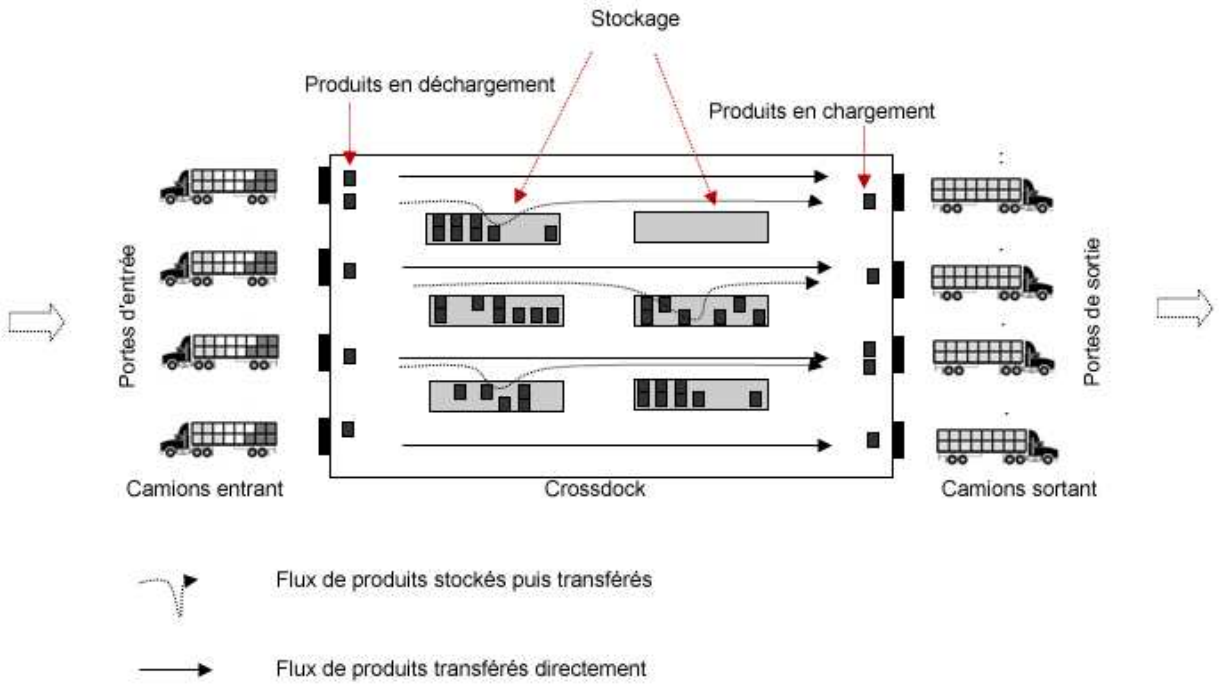

Figure 6.1 – Exemple de “crossdock” (entrepôt de transbordement)

Le *problème de transbordement* peut être défini comme étant le problème de logistique visant à réduire les différents coûts de stockage, de transport, les coûts des opérations de manutention intermédiaire et les délais de livraison correspondant à l'ensemble des opérations permettant l'échange de produits ou de passagers entre deux ou plusieurs moyens de transport transitant par les entrepôt de transbordement.[51].

6.1.2 Applications industrielles

Il existe divers cas d'applications industrielles des problèmes de transbordement. Parmi ces cas, on peut citer le domaine du courrier postal dans lequel le problème prend ses origines. Il existe aussi des exemples d'applications dans les domaines de la distribution en grandes surfaces, la distribution de produits périssables et/ou frais, et l'industrie de distribution de la presse, pour ne citer que ceux-là.

6.1.3 État de l'art

De façon globale, il existe deux types de problématiques autour du transbordement : le niveau stratégique et le niveau opérationnel.

6.1.3.1 Le niveau stratégique

Ce niveau concerne les aspects de *localisation* des centres de transbordement (les crossdocks), leur *taille* et leur *forme*. Ces décisions d'ordre stratégique doivent être prises en toute connaissance de la situation actuelle du marché, de son environnement extérieur et de ses perspectives d'évolution. Elles englobent donc un caractère d'incertitude. Les conséquences d'une mauvaise décision à ce niveau-là, s'étendraient sur toute la chaîne logistique et à long terme. Dans la littérature, il existe des articles tels celui de Klose et Drexl [53] qui décrivent plus en détail l'aspect localisation dans ce type de problème. Pour les aspects de la forme des crossdocks et du nombre de portes, on peut se référer à l'article de Bartoldi et Gue en 2004 [13].

6.1.3.2 Le niveau opérationnel

Un second type de problématique se situe au niveau opérationnel dans la prise des décisions. Il existe en général deux types d'approches pour la résolution de cette partie du problème : la première se focalise sur les opérations interactives entre le crossdock et les autres éléments de la chaîne de distribution ; la seconde est centrée sur l'ordonnancement des opérations de manutention à l'intérieur du crossdock.

6.1.3.2.1 Première approche

Dans la première approche, plus axée sur les opérations à l'extérieur de la zone de stockage, on se focalise sur les interactions entre les crossdocks et les autres organes de la chaîne logistique (du réseau de distribution). Il s'agit principalement des problèmes de minimisation des coûts de transport, du nombre de camions nécessaires et de la planification de leur trajet ([28, 37, 65]).

Il peut s'agir aussi des *problèmes d'affectation* des camions aux différentes portes du crossdock ([92, 93]) de sorte à minimiser les distances parcourues et les coûts de déplacement des produits à l'intérieur de l'entrepôt. On considère alors que grâce à une meilleure affectation des camions aux différentes portes d'entrée et de sortie, on peut faire des économies sur les coûts de manutention. On essaye donc de déterminer l'ordre d'arrivée et l'ordre de départ des camions qui optimisent au mieux le déplacement des produits entre les deux types de portes d'entrée.

6.1.3.2.2 Seconde approche

Contrairement à la première, la seconde approche se focalise plus sur les opérations à l'intérieur de la zone de stockage, sachant que la fonction capitale à l'intérieur d'un entrepôt de transbordement est la consolidation (regroupement et, accessoirement, tri, contrôle et ré-emballage) des produits de même destination et un remplissage optimal des camions sortant du crossdock.

Cette approche consiste à déterminer le meilleur ordonnancement des tâches de manu-

tention (déplacement des produits déchargés aux portes d'entrée et devant être chargés aux portes de départ) et de coût minimal. On parlera alors de *problème d'ordonnancement des tâches de manutention*.

Même si ce problème est traité différemment du RCPSP, que nous avons vu dans la première partie de ce manuscrit, ils se rejoignent bien sûr sur la thématique générale de cette thèse portant sur *l'ordonnancement*. Dans la suite de ce chapitre, nous nous intéressons donc à cette approche du problème.

Formellement, on considère que les produits déchargés aux portes d'entrée peuvent être chargés aux portes de sortie selon trois cheminements (politiques de déchargement/chargement) possibles :

- N'ayant pas le droit d'être stockés, les produits ne sont déchargés que s'ils peuvent être chargés directement dans les camions sortant du site : il s'agit là d'un processus très rigide.
- On décharge tous les produits, on les stocke un court moment, le temps d'effectuer certaines opérations nécessaires à la nature de ces produits, puis on les charge dans les camions sortants : la particularité de ce processus est d'être très coûteux.
- Une troisième politique préconise que les produits sont déchargés, ceux pouvant être chargés directement sont convoyés, les autres étant stockés le temps de pouvoir être chargés : par rapport aux deux précédents, ce processus est plus flexible et moins coûteux à la fois.

Cette problématique reste à ce jour très peu étudiée. Les premières véritables études remontent en 2002, où Wooyeon Yu [97] dans le cadre de ses travaux de thèse, présente ce type de problème contenant une seule porte d'entrée (déchargement) et une seule porte de sortie (chargement). Il teste différentes politiques de déchargement/chargement telles que la première et la troisième politiques décrites ci-dessus. Le critère est la minimisation du temps total des opérations de manutention à l'intérieur du crossdock.

Li *et al.* [66], en 2004, assimilent le problème à un problème à machines parallèles à deux niveaux où les camions en déchargement ou chargement sont considérés comme des tâches, devant être exécutées par les machines (ressources cumulatives). Ici, les machines peuvent être considérées comme les équipes de travail, ce qui lui permet de proposer un modèle basé sur la PLNE et une heuristique fondée sur un algorithme génétique.

Mc Williams *et al.* [71], en 2005, dans leur étude sur le flux de circulation des produits à l'intérieur des crossdocks, étudient à l'aide d'un algorithme génétique et de simulation, un modèle de minimisation de l'intervalle de temps entre les premier et dernier produits déchargés par les camions aux portes de déchargement.

Yu et Egbelu en 2008 [98] proposent 3 modèles pour la résolution de ce problème (problème à une seule entrée et une seule sortie) dont une de type heuristique. Pour les deux autres nécessitant des temps de calcul importants, on y retrouve une formulation de type PLNE et un modèle basé sur une énumération complète de toutes les possibilités entre l'arrivée et le départ.

Dans sa thèse, R. Larbi en 2008 [62] applique différentes méthodes d'optimisation à plusieurs variantes du problème d'optimisation du fonctionnement interne d'une plateforme de transbordement. Notamment, il propose une méthode de recherche tabou pour la résolution du cas à une seule porte d'entrée et une seule porte de sortie, avec l'ordre d'arrivée des camions connu. Comme critère, il choisit de minimiser le coût total (coût de stockage intermédiaire et coût de manutention entre les portes). Des heuristiques sont proposées pour des extensions considérant un ordre d'arrivée des camions inconnu ou partiellement connu ainsi que plusieurs portes d'entrée et de sortie.

6.2 Description du problème traité

Dans notre étude, nous nous sommes intéressés à l'optimisation des opérations de manutention entre les portes. Globalement, on considère qu'un produit déchargé à la

porte d'entrée, peut :

- soit être acheminé directement à une porte de sortie, où il sera directement chargé dans un camion sortant ;
- soit être transféré dans une zone de stockage en attendant d'être chargé.

On considère le cas avec une seule porte d'entrée et une seule porte de sortie. Les ordres d'arrivée et de départ des camions sont connus. Un camion positionné à la porte d'entrée ne la quitte que s'il est totalement déchargé. De même, un camion positionné à la porte de départ ne la quitte que s'il est totalement plein. En d'autres termes, on n'autorise aucune rotation des camions aux différentes portes.

Les produits sont identifiés et sont chargés dans les camions en fonction de leur destination finale et un camion à la porte de départ ne part que dans une et seule destination finale. Ainsi, un produit déchargé pourra être directement transféré dans le camion se trouvant à la porte de départ si ce camion correspond à la destination finale du produit. Dans le cas contraire, il est mis en attente dans la zone de stockage temporaire. En outre, on considère que la capacité de la zone de stockage est illimitée.

Tous les camions, tant ceux accostant à la porte d'entrée que ceux positionnés à la porte de sortie, ont la même capacité. De ce fait, pour être sûr de pouvoir transférer et charger tous les produits déchargés, on admet que le nombre de camions entrants est égal au nombre de camions sortants.

La politique de déchargement des produits peut influencer les opérations de manutention, en influençant le nombre de produits stockés temporairement (appelé *déposes*). Comme politiques de déchargement des produits, on a :

- **FIFO (***First In First Out***)** : Le premier produit chargé dans le camion sera le premier à être déchargé.
- **LIFO (***Last In First Out***)** : Le dernier produit chargé dans le camion sera le premier à être déchargé.

– **Le déchargement libre** : n'importe quel produit du camion peut être déchargé dans n'importe quel ordre. Ainsi, tous les produits du camion peuvent être déchargés sans aucune restriction. Dans le cadre de notre étude, nous avons retenu cette politique de déchargement.

Parallèlement, on peut distinguer deux types de politiques extrêmes de chargement. Ainsi, pour un camion se trouvant à la porte de sortie allant dans une direction pour laquelle des produits se trouvent à la fois dans la zone de stockage et dans le camion se trouvant à la porte d'entrée :

– on charge a priori les produits se trouvant dans la zone de stockage temporaire ;
– ou au contraire, on ne charge uniquement que les produits se trouvant dans les camions (celui se trouvant à la porte d'entrée, mais aussi ceux en attente qui se présenteront à la porte d'entrée dès que le camion précédant en sera parti). Les produits se trouvant dans la zone de stockage ne sont chargés que lorsqu'il ne restera plus de produits de cette destination dans aucun camion entrant.

Entre ces deux politiques extrêmes, il existe une politique intermédiaire, qui consiste à autoriser le modèle à choisir à chaque étape, de charger soit une unité produit en stock, soit une unité de produit disponible dans les camions. Dans notre étude, nous n'avons retenu que cette configuration intermédiaire.

6.3 Résolution du problème par branch-and-bound

Dans cette approche, nous avons dans un premier temps construit un réseau de possibilités, N étant le nombre de camions entrants (équivalent aussi au nombre de camions sortants). Il s'agit d'un réseau de $N \times N$ nœuds, où chaque nœud, représentant un état du système, est identifié par une paire (i, o), avec i représentant l'ordre du camion entrant et o celui du camion sortant. À chaque nœud, pour le camion se trouvant à la porte d'entrée, on mémorise les informations concernant le nombre de produits transférés directement dans le camion à la porte de sortie et le nombre de produits déposés dans la zone de stockage.

À partir d'un état donné (i,o), on ne peut évoluer que dans l'un des deux états suivants $(i+1,o)$ et $(i,o+1)$, traduisant des informations différentes :

- État $(i+1,o)$: La camion *i* se trouvant à la porte d'entrée est totalement déchargé et est remplacé par le camion $i+1$. Il s'agit donc de l'arrivée d'un autre camion à la porte de déchargement (porte d'entrée).
- État $(i,o+1)$: La camion *o* se trouvant à la porte de sortie est totalement plein et est remplacé par le camion $o+1$. Ici, il s'agit plutôt du départ d'un camion plein en direction d'une destination finale d'ensemble de produits.

Partant donc de ce principe, toute *solution du système* correspond à un ensemble de déplacements commençant par le nœud $(1,1)$ et se terminant par le nœud (N,N). Une solution optimale consistant à minimiser le nombre de déposes, cela revient à maximiser le nombre de produits transférés directement de la porte d'entrée à la porte de sortie.

Nous proposons une résolution de type branch-and-bound que nous décrivons ci-dessous.

Branch-and-bound proposé

Pour la conception de la procédure, on note que la construction d'une solution ne respecte pas la propriété de Markov. C'est-à-dire que la valeur de la solution courante ne dépend pas uniquement de l'état précédent, mais plutôt de tous les états précédents, depuis l'état initial $(1,1)$. Par conséquent, pour trouver la solution optimale il est nécessaire de brancher sur tous les chemins intermédiaires, sachant que la valeur de chaque nœud peut être calculée facilement en un temps assez court.

Typiquement, notre algorithme se présente de la manière suivante :

- La première étape dans cet algorithme après la construction du réseau des états est la détermination des solutions réalisables. Compte tenu des hypothèses évoquées plus haut, certains nœuds du réseau ne pourront jamais être utilisés. Par exemple,

on ne pourra jamais atteindre le nœud $(1,2)$, car pour charger le camion 2 au quai de départ, il faudrait au préalable charger le camion 1. Or, tous les camions étant de même capacité, le camion 1 au quai d'arrivée ne pourra jamais disposer de suffisamment de produits pour remplir le camion 1 du quai de départ et en mettre à la fois dans le camion suivant, sachant qu'à l'état initial $(1,1)$, on ne dispose d'aucun produit au sol (en zone de stockage) (voir figure 6.2 ci-dessous).

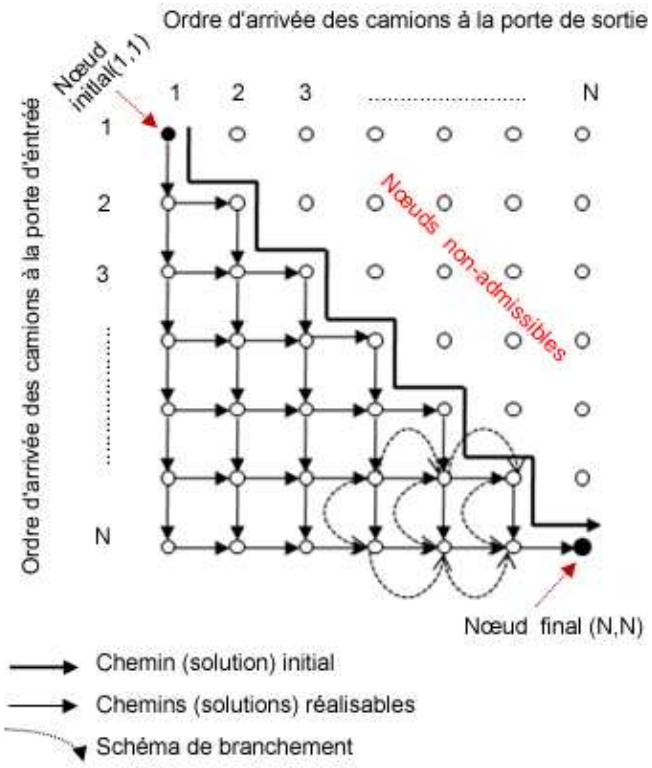

Figure 6.2 – Réseau des états et espace de solutions réalisables

– Dans la seconde étape, on exécute une recherche de type profondeur d'abord pour déterminer la solution initiale, selon la politique suivante : On favorise en priorité le remplissage du camion se trouvant au quai de sortie. Par conséquent, dans la construction de cette première solution, on n'hésitera pas à utiliser les produits se trouvant dans la zone de stockage, s'il n'existe plus de produits de cette destination dans le camion se trouvant à la porte d'entrée.
– Après l'obtention de la solution initiale, en remontant à partir du nœud terminal

(N,N), l'algorithme branche sur tous les chemins alternatifs en explorant dans sa procédure de backtracking jusqu'à atteindre le nœud initial $(1,1)$.

Dans sa procédure de branchement, avant de considérer un nœud, l'algorithme vérifie s'il n'est pas dominé, selon les règles de dominance établies.

La règle de dominance utilise la condition d'optimalité de Pareto. Pour chaque nœud (i,o), on construit une frontière de Pareto $\mathscr{P}(i,o)$ à partir des deux critères suivants :

- $ValP(i,o,k)$: la valeur de la fonction-objectif pour le point k de la frontière de Pareto $P(i,o)$ du nœud (i,o).
- $LoadP(i,o,k)$: le nombre de produits à transférer directement dans le camion se trouvant à la porte de sortie, pour le point k de la frontière de Pareto $\mathscr{P}(i,o)$ du nœud (i,o).

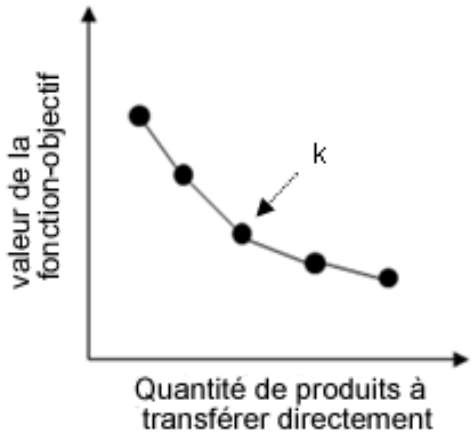

Figure 6.3 – Frontière de Pareto

Le chemin étudié sera dit dominé s'il est dominé par la frontière de Pareto, et donc ne sera pas exploré. Typiquement, pour un nœud (i,o), $Val(i,o)$ la valeur de la fonction-objectif à ce nœud et $Load(i,o)$ le nombre de produits à transférer directement, si $\{Val(i,o) < ValP(i,o,k) \quad \forall k \in \mathscr{P}(i,o)\}$ et $\{Load(i,o) < LoadP(i,o,k) \quad \forall k \in \mathscr{P}(i,o)\}$, alors on dira que ce nœud (i,o) est *Pareto-dominé* et ne sera donc pas exploré.

Dans tous les autres cas, le nœud n'étant pas dominé, il sera testé et ses valeurs permettront de faire une mise à jour des valeurs de la frontière de Pareto pour ce nœud.

6.4 Résultats

À la suite d'une série de tests effectués sur un ordinateur doté d'un processeur Pentium cadencé à 2.2 Ghz, nous avons comparé les résultats de cette méthode à une résolution par la programmation dynamique [69].
Nous avons généré plusieurs groupes d'instances, avec un nombre de camions variant d'un groupe à l'autre. Ainsi, on a des instances de 5, 10, 20, 40, 80, 160, 320 et 640 camions. Cependant, le nombre de destinations possibles reste fixé à 5. Chaque groupe d'instances comporte 20 instances.

Nous avons comparé le temps moyen mis pour la résolution de chaque groupe d'instances. Les résultats obtenus sont présentés dans le tableau 6.I. Ces résultats montrent

Tableau 6.I – Temps moyen de résolution (en seconde)

Taille des instances	Algorithmes	
	Prog. dynamique	Branch-and-bound
5	10^{-2}	10^{-4}
10	1.1×10^{-2}	10^{-4}
20	3.51×10^{-2}	10^{-3}
40	7.65	8×10^{-3}
80	8.75×10^{1}	6.16×10^{-2}
160	3.45×10^{2}	4.37×10^{-1}
320	1.41×10^{3}	4.07
640	7.85×10^{3}	5.40×10^{1}

une domination de notre modèle sur toutes les instances. De plus, pour le traitement des instances de tailles de plus en plus grandes, tandis que le temps de calcul pour la résolution par la programmation dynamique augmente rapidement, celui obtenu par notre modèle reste relativement faible (cf. figure 6.4). Par exemple, pour les instances de 640 camions, il faut en moyenne plus de deux heures, alors que notre modèle met moins d'une minute.

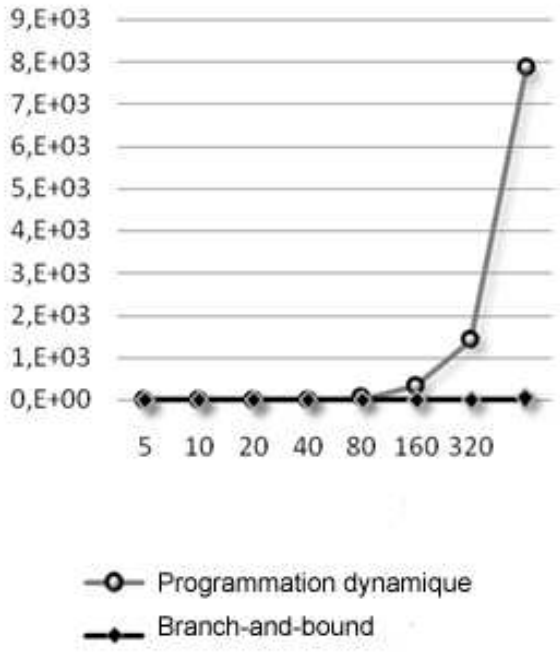

Figure 6.4 – Courbes de comparaison des temps de résolution

6.5 Conclusion

Dans ce chapitre, nous avons étudié le problème de transbordement à une seule porte d'entrée et une seule porte de sortie, dans son aspect qui consiste ordonnancer les opérations de manutention entre les portes.

Nous avons proposé une résolution par un modèle de type branch-and-bound, avec un critère de dominance basée sur l'examination de la frontière de Pareto. Comparé à une résolution par la programmation dynamique qui avait déjà été appliquée sur ce même problème, nous obtenons les meilleurs temps de résolution. La complexité de ce problème n'étant pas prouvée, la forme quasi-linéaire de la courbe du temps de résolution par notre branch-and-bound, accentue notre intuition qu'il soit polynomial, ce qui reste encore à démontrer ou infirmer.

Les résultats de ces travaux posent clairement le problème de la complexité du problème. Car, même si nous avions de forts soupçons sur le caractère polynomial de la

complexité, celle-ci ne semblait pas évidente à montrer (à cause du nombre d'états du système qui peut être important).

Cependant, la publication de nos résultats concernant nos travaux à inspiré certains auteurs, notamment Ruslan Sadykov [87] en septembre 2009, donc tout récemment, qui démontre clairement que le problème est polynomial. La plus grande contribution de ce branch-and-bound est qu'il a permis de lever le doute et stimuler la recherche sur la démonstration de la polynomialité de problème. Depuis lors, même si elle n'a pas encore publié ces résultats, l'équipe montréalaise avec laquelle pour avons réalisé ces travaux, démontre également la polynomialité du problème.

En perspective de ce travail ponctuel sur le transbordement, une extension de ce modèle au problème à plusieurs portes d'entrée et plusieurs portes de sortie serait envisageable. Il peut servir de support pour l'évaluation de l'influence des ordres d'arrivée et départ sur les performances de fonctionnement d'un crossdock.

CONCLUSION GÉNÉRALE & PERSPECTIVES

Dans ce travail de thèse, nous avons étudié deux types de problèmes d'ordonnancement. La majeure partie concerne le problème d'ordonnancement de projet à moyens limités (RCPSP). Le problème d'ordonnancement des opérations de manutention dans un entrepôt de transbordement ("crossdocking") est également traité avec une moindre importance.

Dans une première partie (clairement la plus étendue donc), nous abordons le RCPSP. À partir de modélisations utilisant la programmation linéaire en nombres entiers, nous avons proposé deux nouvelles formulations de ce problème, utilisant des variables indicées par des événements. Dans l'une d'entre elles, on utilise une variable binaire pour marquer le début de l'exécution de chaque activité et une autre variable pour marquer sa fin. Dans la seconde proposition, une seule variable par événement est utilisée. Elle identifie les événements après lesquels l'activité reste en cours ou débute son exécution.

De façon générale, comparées à d'autres modèles de la littérature sur divers types d'instances, nos propositions affichent des résultats plus intéressants sur les instances contenant des activités aux durées disparates et associées à de longs horizons d'ordonnancement. En particulier, sur ces mêmes types d'instances mais hautement cumulatives (caractéristiques de base du RCPSP), elles sont également les plus performantes.

Nous avons également abordé la résolution d'une extension du RCPSP consistant à prendre en compte des ressources particulières, qui peuvent être consommées en début d'exécution de chaque activité, mais aussi produites à leur fin : il s'agit du RCPSP avec consommation et production de ressources.
Afin d'effectuer une comparaison expérimentale entre différents modèles, nous avons proposé une adaptation de nos formulations basées événements, des formulations à temps discret de Pritsker et de Christofides, et de la formulation à temps continu basée sur les flots (proposé par Artigues sur la base des travaux de Balas). Globalement, les résultats

montrent que nos formulations basées événements obtiennent les meilleurs résultats sur de nombreux types d'instances.

Dans le dernier chapitre de ce document, nous proposons un branch-and-bound utilisant des coupes basées sur la *frontière de Pareto*, pour la résolution du problème d'ordonnancement des opérations de manutention au sein d'un entrepôt de transbordement ("crossdocking"). Les excellents résultats obtenus ont renforcé nos interrogations sur la complexité non-prouvée de ce problème, qui, depuis, s'est effectivement révélée polynomiale.

En perspective pour la suite de ce travail, notamment sur le RCPSP, il nous paraît particulièrement intéressant de travailler sur l'élaboration de coupes performantes agissant sur la réduction du nombre d'événements de nos modèles. L'intérêt de cette piste est corroboré par des résultats préliminaires que nous avons obtenus sur une variante d'un de nos modèles à événements. Ceci aurait sûrement pour effet d'améliorer les performances de nos propositions et d'envisager leur suprématie (dans tous les cas de figures) face aux formulations proposées dans la littérature.

Il est enfin envisagé de travailler sur des modèles hybrides de nos propositions, incluant ainsi des techniques telles que la programmation par contraintes ou des techniques de type métaheuristique.

BIBLIOGRAPHIE

[1] www.laas.fr/laas/MOGISA/RPCSP-instances/high_duration_range.tar.gz.

[2] M. K. Acar. Robust dock assignments at less-than-truckload terminals. Master's thesis, University of South Florida, 2004.

[3] R. Alvarez-Valdès et J. M. Tamarit. The project scheduling polyhedron : dimension, facets and lifting theorems. *European Journal of Operational Research*, 67 (2):204–220, 1993.

[4] D. Applegate et W. Cook. A computational study of job-shop scheduling. *ORSA Journal on Computing*, 3(2):149–156, 1991.

[5] Artigues, O. Koné, P. Lopez et M. Mongeau. Event-based MILP models for resource-constrained project scheduling problems. *EURO'09*, July 2009. Bonn, Allemagne.

[6] C. Artigues, O. Koné, P. Lopez, M. Mongeau, E. Néron et D. Rivreau. Computational experiments. Dans C. Artigues, S. Demassey et E. Néron, éditeurs, *Resource-Constrained Project Scheduling Models, algorithms, extensions and applications*, pages 98–102. ISTE/Wiley, 2008.

[7] C. Artigues, P. Michelon et S. Reusser. Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research*, 149(2):249–267, 2003.

[8] T. Baar, P. Brucker et S. Knust. *Meta-heuristics : advances and trends in local search paradigms for optimization*, chaptitre Tabou-search algorithms and bounds for the resource-constrained project scheduling problem, pages 1–18. Kluwer, 1998.

[9] E. Balas. *Applications of mathematical programming techniques*, chaptitre Project scheduling with resource constraints, pages 187–200. American Elsevier, 1970.

[10] P. Baptiste et C. Le Pape. Constraint propagation and decomposition techniques for highly disjunctive and highly cumulative project scheduling problems. *Constraints*, 5(1-2):119–139, 2000.

[11] P. Baptiste, C. Le Pape et W. Nuijten. Satisfiability tests and time-bound adjustments for cumulative scheduling problems. *Annals of Operations Research*, 92: 305–333, 1999.

[12] J. J. Bartholdi et K. R. Gue. Reducing labor costs in an LTL crossdocking terminal. *Operational research*, 48(6):823–832, 2002.

[13] J. J. Bartholdi et K. R. Gue. The best shape for a crossdocking. *Transportation Science*, 2:235–244, 38.

[14] J. C. Beck. Heuristics for constraint-discreted scheduling with inventory. *Journal of Scheduling*, 5:45–69, 2002.

[15] R. Bermudez et M. H. Cole. A genetic algorithm approach to door assignments in breakbulk terminals. Technical report, University of Arkansas, Fayette Ville USA, 2002.

[16] J. Blazewicz, J. Lenstra et A. Rinnooy Kan. Scheduling subject to resource constraints : Classication and complexity. *Discrete Applied Mathematics*, 5(1): 11–24, 1983.

[17] F. F. Boctor. Resource-constrained project scheduling by simulated annealing. *International Journal of Production Research*, 34(8):2335–2361, 1996.

[18] K. Bouleimen et H. Lecoq. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research*, 149(2):268–281, 2003.

[19] H. Bouly, J. Carlier, A. Moukrim et M. Russo. Solving RCPSP with resources production possibility by tasks. *In : MHOSI'2005, 24–26 Avril*, 2005.

[20] A. M. Brown. *Improving the efficiency of hub Operations in less-than-truckload distribution network*. Phd thesis, Virginia Polytechnic Institute, 2003.

[21] P. Brucker. Schudeling and constraint propagation. *Discrete Applied to Mathematics*, 123(1–3):227–256, 2002.

[22] P. Brucker et S. Knust. A linear programming and constraint propagation-based lower bound for the RCPSP. *European Journal of Operational Research*, 127: 355–362, 2000.

[23] P. Brucker, S. Knust, A. Schoo et O. Thiele. A branch and bound algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 107:272–288, 1998.

[24] J. Carlier. The one-machine sequencing problem. *European Journal of Operational Research*, 11(1):42–47, 1982.

[25] J. Carlier et B. Latapie. Une méthode arborescente pour résoudre les problèmes cumulatifs. *RAIRO – Recherche opérationnelle*, 25(3), 1991.

[26] J. Carlier, A. Moukrim et H. Xu. The project scheduling problem with production and consumption of resources : A list-scheduling based algorithm. *Discrete Applied Mathematics*, 2009.

[27] J. Carlier et E. Néron. On linear lower bounds for resource constrained project scheduling problem. *European Journal of Operational Research*, 149:314–324, 2003.

[28] P. Chen, Y. Guo et B. Rodrigues. Multiple crossdocks with inventory and time windows. *Computers and of Operations Research*, 33:43–46, 2006.

[29] J.-H. Cho et Y.-D. Kim. A simulated annealing algorithm for the resource-constrained project scheduling problems. *Journal of the Operational Research Society*, 48:736–744, 1997.

[30] N. Christofides, R. Alvarez-Valdès et J. M. Tamarit. Project scheduling with resource constraints : A branch and bound approach. *European Journal of Operational Research*, 29(3):262–273, 1987.

[31] J. Damay. *Techniques de résolution basées sur la Programmation Linéaire pour l'ordonnancement de Projet*. Thèse de doctorat, Université B. Pascal de Clermont-Ferrand, 2005.

[32] J. Damay, A. Quilliot et E. Sanlaville. Linear programming based algorithms for preemptive and non-preemptive RCPSP. *European Journal of Operational Research*, à paraître, 2006.

[33] S. Dauzère-Pérès et J. B. Lasserre. A new mixed-integer formulation of the flow-shop sequencing problem. *2nd Workshop on models and algorithms for planning and scheduling problems, Wernigerode, Allemagne,* may 1995.

[34] S. Demassey. *Méthodes hybrides de programmation par contraintes et de programmation linéaire pour le problème d'ordonnancement de projet à contraintes de ressources*. Thèse de doctorat, Université d'Avignon, 2003.

[35] S. Demassey, C. Artigues et P. Michelon. Constraint propagation based cutting planes : an application to the resource-constrained project scheduling problem. *INFORMS Journal on Computing*, 17(1):52–65, 2005.

[36] E. Demeulemeester et W. Herroelen. New benchmark results for the resource-constrained project scheduling problem. *Management Science,* 43(11):1485–1492, 1997.

[37] H. Donladson, E. L. Johnson, H. D. Ratliff et M. Zhang. Schedule-driven cross-docking netwwork. Technical report, Georgia Institute of Technology, 1998.

[38] S. Elmaghraby et S. E. W. Herroelen. On the measurment of complexity in activity networks. *European Journal of Operational Research*, 5(4):223–234, 1980.

[39] T. Garaix, C. Artigues et S. Demassey. Bornes basées sur les ensembles interdits pour le problème d'ordonnancement de projet à moyens limités. *6ème congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'05), Tours*, 2005.

[40] M. Garey et D. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, 4(4):397–441, 1975.

[41] M. Garey et D. Johnson. *Computers and intractability. A guide to the theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[42] L. Girard et E. Pinson. List scheduling in connexion with tabu search for solving scheduling problems. *Workshop on Project Management and Scheduling (WPMS'98) Istanbul Turquie*, 1998.

[43] K.R. Gue. The effects of trailer scheduling on the layout of freight terminals. *Transportation Science*, 33(4):419–428, 1999.

[44] S. Hartmann. A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics*, 49:433–448, 2002.

[45] W. Herroelen. Project scheduling theory and practice. *Production and Operations Management*, 14(4):413–432, 2006.

[46] W. Herroelen et B. De Reyck. Phase transitions in project scheduling. *Journal of the Operational Research Society*, 150:148–156, 1999.

[47] V. Jayaraman et A. Ross. A simulated annealing methodology to distribution network design and management. *European Journal of Operational Research*, 144(3): 629–645, 2003.

[48] T. Johnson. *An algorithm for the resource-constrained project scheduling problem*. Thèse de doctorat, M.I.T Boston USA, 1967.

[49] J. Jozefowska et J. Weglarz. *Perspectives in modern project scheduling*. Springer, 2006.

[50] L. Kaplan. *Ressource-constrained project scheduling with preemption of jobs*. Thèse de doctorat, University of Michigan, États-Unis, 1988.

[51] E. Kinnear. Is there any magic in cross-docking ? *Supply Chain Management : An International Journal*, 2:49–52, 1997.

[52] R. Klein. Project scheduling with with time-varying resource constraints. *International Journal of Production Research*, 38(16):3937–3952, 2000.

[53] A. Klose et A. Drexl. Facility location models for distribution system design. *European Journal of Operational Research*, 162(1):4–29, 2005.

[54] Y. Kochtetov et A. Stolyar. Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem. *Proceedings of the 3rd International Workshop of Computer Science and Information Technologies*, 2003.

[55] R. Kolisch. Serial and parallel resource-constrained project scheduling methods revisited : Theory and computation. *European Journal of Operational Research*, 90(2):320–333, 1996.

[56] R. Kolisch et A. Sprecher. PSPLIB - A project scheduling library. *European Journal of Operational Research*, 96(1):205–216, 1997.

[57] O. Koné, C. Artigues, P. Lopez et M. Mongeau. Nouvelle formulation du problème d'ordonnancement de projet à moyens limités basée sur les événements. *ROADEF'08*, Février 2008. Clermont-Ferrand.

[58] O. Koné, C. Artigues, P. Lopez et M. Mongeau. Formulation on/off pour le RCPSP. *ROADEF'09*, Février 2009. Nancy.

[59] O. Koné, C. Artigues, P. Lopez et M. Mongeau. Event-based MILP models for resource-constrained project scheduling problems. *Computers & Operations Research*, Soumis.

[60] P. Laborie. Complete MCS-based search : Application to resource constrained project scheduling. *IJCAI*, 2005, 181–186.

[61] P. Laborie. Algorithms for propagating resource constraints in a planning and scheduling : Existing approaches and new results. *Artificial Intelligence*, 143:151–188, 2003.

[62] R. Larbi. *Optimisation des séquences d'opérations dans une plateforme de cross-docking*. Thèse de doctorat, Institut Polytechnique de Grenoble, 2008.

[63] J. B. Lasserre et M. Queyranne. Generic scheduling polyhedra and a new mixed-integer formulation for single-machine scheduling, integer programming and combinatorial optimization. *Proceedings of the 2nd International IPCO Conference*, pages 136–149, 1992.

[64] J.-K. Lee et Y.-K. Kim. Search heuristics for resource constrained project scheduling. *Journal of the Operational Research Society*, 47:678–689, 1996.

[65] Y. H. Lee, J. W. Jung et K. M. Lee. Vehicle routing scheduling for cross-docking in the splly chain. *Computers and Industrial Engineering*, 51:247–256, 2006.

[66] Y. Li, A. Lim et B. Rodrigues. Crossdocking JIT scheduling with times windows. *Journal of the Operational Research Society*, 55:1342–1351, 2004.

[67] I. Lizarralde. *Aide au pilotage d'activités d'ingénierie pour le développement distribué d'un système complexe*. Thèse de doctorat, Institut National des Sciences Appliquées de Toulouse, 2007.

[68] P. Lopez. *Approche énergétique pour l'ordonnancement de tâches sous contraintes de temps et de ressources*. Thèse de doctorat, Université Paul Sabatier, Toulouse, 1991.

[69] Y. Maknoon, P. Baptiste et O. Koné. Optimal loading and unloading policy in cross-docking platform. *13th Symposium on Information Control Problems in Manufacturing (INCOM'09), Moscou, Russie*, june 3-5 2009.

[70] A. Mastor. An experimental and comparative evaluation of production line balancing techniques. *Management Science*, 16(11):728–746, 1970.

[71] D. McWilliams, P. M. Stanfield et C. D. Geiger. The parcel hub scheduling problem : A simulation-based solution approach. *Computers and Industrial Engineering*, 49:393–412, 2005.

[72] D. Merkle, M. Middendorf et H. Schmeck. Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation*, 6:333–346, 2002.

[73] A. Mingozzi, V. Maniezzo, S. Ricciardelli et L. Bianco. An exact algorithm for the multiple resource-constrained project scheduling problem based on a new mathematical formulation. *Management Science*, 44(5):714–729, 1998.

[74] R. H. Möhring, A. S. Schulz, F. Stork et M. Uetz. Solving project scheduling problems by minimum cut computations. *Management Science*, 49:330–350, 2003.

[75] K. Neumann et C. Schwindt. Project scheduling with inventory constraints. *Mathematical Methods of Operations Research*, 56:513–533, 2002.

[76] E. Néron. *Du flow-shop hybride au problème cumulatif*. Thèse de doctorat, Université de Technologie de Compiègne, 1999.

[77] E. Néron, C. Artigues, P. Baptiste, J. Carlier, S. Demassey et P. Laborie. *Perspectives in Modern Project Scheduling*, volume 92 de *International Series in Operations Research & Management Science*, chaptitre Lower bounds computation for RCPSP. Józefowska and Joanna and Weglarz and Jan (Eds.)„ springer édition, 2006.

[78] Y. Oh, H. Hwang, C.N. Cha et S. Lee. Dock-door assignment problem for the korean mail distribution center. *Computers and Industrial Engineering*, 51(2):288–296, 2006.

[79] M. Palpant, C. Artigues et P. Michelon. LSSPER : Solving the resource-constrained project scheduling problem with large neighbourhood search. *Annals of Operations Research*, 131(1–4):237–257, October 2004.

[80] J. Patterson, R. Slowinski, F. Talbot et J. Weglarz. Computational experience with a backtracking algorithm for solving a general class of precedence and resource constrained scheduling problems. *European Journal of Operational Research*, 49: 68–79, 1990.

[81] E. Pinson, C. Prins et F. Rullier. Using tabu search for solving the resource constrained project scheduling problem. *Proceedings of the Fourth International Workshop on Project Management and Scheduling*, pages 102–106, 1994.

[82] J. M. Pinto et I. E. Grossmann. A continuous time MILP model for short term scheduling of batch plants with pre-ordering constraints. *Industrial & Engineering Chemistry Research*, 3037–3051, 1995.

[83] A. Pritsker, L. Watters et P. Wolfe. Multi-project scheduling with limited resources : A zero-one programming approach. *Management Science*, 16:93–108, 1969.

[84] A. Sprecher R. Kolisch et A. Drexl. Characterization and generation of a general class of resource-constrainted project scheduling problems. *Management Science*, 41:1693–1703, 1995.

[85] H. D. Ratliff, J. V. Vate et M. Zhang. Network design for load-driven cross-docking systems. Technical report, Georgia Institute of Technology, 1998.

[86] B. De Reyck et W. Herroelen. On the use of the complexity index as a measure of complexity in activity networks. *European Journal of Operational Research*, 91 (2):347–366, 1996.

[87] Ruslan Sadykov. A polynomial algorithm for a simple scheduling problem at cross docking terminals. Research report, INRIA, September 2009.

[88] J. K. Sankaran, D. L. Bricker et S.-H Juang. A strong frationnal cutting-plane algoritm for resource-constrained project scheduling. *International Journal of Industrial Engineering : Applications and Practice*, 6(2):99–111, 1999.

[89] A. Sprecher. Scheduling resource-constrained projects competitively at modest memory requirements. *Management Science*, 46:710–723, 2000.

[90] A. Sprecher, R. Kolisch et A. Drexl. Semi-active, active, and non-delay schedules for resource-constrained project scheduling problem. *European Journal of Operational Research*, 80:94–102, 1995.

[91] J. P. Stinson, E. W. Davis et B. M. Khumawala. Multiple resource-constrained scheduling using branch-and-bound. *AIIE Transactions*, 10(3):252–259, 1978.

[92] L. Y. Tsui et C. H. Chang. A microcomputer based decision support tool for assigning dock doors in freight yards. *Computers and Industrial Engineering*, 19: 309–312, 1990.

[93] L. Y. Tsui et C. H. Chang. Optimal solution to a dock assignment problem. *Computers and Industrial Engineering*, 19:283–286, 1992.

[94] M. Uetz. *Algorithms for Deterministic and Stochastic Scheduling*. Thèse de doctorat, Technische Universität Berlin, 2001.

[95] V. Valls, M. Qiuinanille et F. Ballestin. Resource-constrained project scheduling : A critical reordering heuristic. *European Journal of Operational Research*, 149: 282–301, 2003.

[96] L. Wolsey. *Integer Programming*. Wiley, 1998.

[97] W. Yu. *Operational Strategies for Cross Docking Systems*. Thèse de doctorat, Iowa State University, 2002.

[98] W. Yu et P.J. Egbelu. Scheduling of inbound and outbound trucks in cross docking systems with temporary storage. *European Journal of Operational Research*, 184 (1):377–396, 2008.

[99] J. C. Zapata, B. M. Hodge et G. V. Reklaitis. The multimode resource constrained multiproject scheduling problem : Alternative formulations. *AIChE Journal*, 54(8): 2101–2119, 2008.

Nouvelles approches pour la résolution du problème d'ordonnancement de projet à moyens limités

RESUME

Dans ce travail de thèse, nous avons étudié deux types de problèmes d'ordonnancement. La majeure partie concerne le problème d'ordonnancement de projet à moyens limités (RCPSP). Le problème d'ordonnancement des opérations de manutention dans un entrepôt de transbordement ("crossdocking") est également traité avec une moindre importance.

Dans une première partie (la plus étendue), nous abordons le RCPSP. À partir de modélisations utilisant la programmation linéaire en nombres entiers, nous avons proposé deux nouvelles formulations de ce problème, utilisant des variables indicées par des événements. Dans l'une d'entre elles, on utilise une variable binaire pour marquer le début de l'exécution de chaque activité et une autre variable pour marquer sa fin. Dans la seconde proposition, une seule variable est utilisée. Elle identifie les événements après lesquels l'activité reste en cours ou débute son exécution. De façon générale, comparées à d'autres modèles de la littérature sur divers types d'instances, nos propositions affichent des résultats plus intéressants sur les instances contenant des activités aux durées disparates et associées à de longs horizons d'ordonnancement. En particulier, sur ces mêmes types d'instances mais hautement cumulatives (caractéristiques de base du RCPSP), elles sont également les plus performantes.

Nous avons également abordé la résolution d'une extension du RCPSP consistant à prendre en compte des ressources particulières, qui peuvent être consommées en début d'exécution de chaque activité, mais aussi produites à leur fin : il s'agit du RCPSP avec consommation et production de ressources. Afin d'effectuer une comparaison expérimentale entre différents modèles, nous avons proposé une adaptation de nos formulations basées événements, des formulations à temps discret de Pritsker et de Christofides, et de la formulation à temps continu basée sur les flots (proposé par Artigues sur la base des travaux de Balas). Globalement, les résultats montrent que nos formulations basées événements obtiennent les meilleurs résultats sur bon nombre de types d'instances.

Dans la seconde partie (plus réduite), nous avons également proposé un branch-and-bound utilisant des coupes basées sur la frontière de Pareto, pour la résolution du problème d'ordonnancement des opérations de manutention au sein d'un entrepôt de transbordement ("crossdocking"). Les excellents résultats obtenus ont renforcé nos interrogations sur la complexité non-prouvée de ce problème, et ont permis d'établir par la suite que le problème est de complexité polynomiale.

New approaches for solving the Resource-Constrained Project Scheduling Problem

ABSTRACT

In this thesis, we studied two types of scheduling problems. The major part concerns the Resource-Constrained Project Scheduling Problem (RCPSP). The scheduling problem of handling operations in a warehouse of Crossdocking is also dealt.

First, from models using mixed integer linear programming, we proposed two new formulations of this problem, using variables indexed by events. In one of them, we use a binary variable to mark the beginning of the performing of each activity and another variable to mark its end. In the second proposal, a single variable is used. It identifies events in which the activity starts or continues its performing. Overall, compared to other models in the literature on various types of instances, our proposals show more interesting results on the instances with long scheduling horizons containing activities with disparate durations. In particular, on the highly cumulative instances (basic characteristics of RCPSP) of these types of instances, they are the most efficient.

We also treat the resolution of the extension of the RCPSP which consists in taking into account specific resources that can be consumed during the performing of each activity, but also produced in another quantity at the end of performing of each activity: it is the RCPSP with consumption and production of resources. To make a comparison between different experimental models, we proposed an adaptation of our event-based formulations, the discrete-time formulations of Pritsker and Christofides, and the flow-based continuous-time formulation (proposed by Artigues on basis of the work of Balas). Overall, the results show that our event-based formulations are most successful on many types of instances.

Second, in one less extensive part, we proposed a branch-and-bound using some cuts based on the Pareto frontier for the resolution of the scheduling problem of handling operations in a warehouse of Crossdocking. The excellent results obtained, which had strengthened our questions about the non-proved complexity of this problem, have contributed to establish later that this problem is of polynomial complexity.