Planification et ordonnancement de projets sous contraintes de ressources complexes

Pierre-Antoine Morin

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *6 décembre 2018* **par**

Pierre-Antoine MORIN

Planification et ordonnancement de projets sous contraintes de ressources complexes

---

JURY

Christian ARTIGUES	Directeur de Recherche CNRS	LAAS-CNRS, Toulouse
Jean-Charles BILLAUT	Professeur des Universités	Université de Tours
Cyril BRIAND	Professeur des Universités	Université de Toulouse III
François CLAUTIAUX	Professeur des Universités	Université de Bordeaux
Sophie DEMASSEY	Maître Assistante	Mines ParisTech, Sophia Antipolis
Alain HAÏT	Professeur	ISAE-SUPAERO, Toulouse
Norbert TRAUTMANN	Professeur	Université de Berne

École doctorale et spécialité :
MITT : Signal, Image, Acoustique et Optimisation

Unités de Recherche :
ISAE SUPAERO / DISC : Département d'Ingénierie des Systèmes Complexes
LAAS CNRS / ROC : Recherche Opérationnelle, Optimisation Combinatoire et Contraintes

Directeurs de Thèse :
Alain HAÏT et *Christian ARTIGUES*

Rapporteurs :
Sophie DEMASSEY et *Norbert TRAUTMANN*

Planification et ordonnancement de projets sous contraintes de ressources complexes

Pierre-Antoine Morin

Thèse soutenue le 6 décembre 2018

Remerciements

En premier lieu, je tiens à remercier chaleureusement mes deux directeurs de thèse, M. Alain Haït et M. Christian Artigues, tant pour leur qualités professionnelles que pour leurs qualités humaines. C'était pour moi un immense honneur de travailler aux côtés d'experts dans le domaine de la gestion de projets sous contraintes de ressources, et de manière plus générale de la recherche opérationnelle. Ils ont su me guider et me supporter avec ardeur et constance tout au long de ces trois dernières années. Je les remercie également pour les opportunités de rencontres scientifiques extrêmement enrichissantes dont ils m'ont fait bénéficier.

À ce titre, je tiens à remercier le professeur Frits Spieksma pour l'intérêt qu'il a porté à mes travaux, m'amenant ainsi à me questionner sur la nature profonde du problème principal étudié au cours de ma thèse. Je le remercie également de m'avoir accueilli au sein de la Faculty of Economics and Business de KU Leuven, ainsi que tous les membres de l'équipe ORSTAT. Je souhaite aussi remercier le professeur Tamás Kis pour les échanges fructueux que nous avons pu avoir, tant sur des aspects de complexité que de modélisation.

Je tiens à remercier Mme Sophie Demassey et M. Norbert Trautmann pour le temps qu'ils ont consacré à la lecture de mon manuscrit et pour la qualité de leurs critiques à la fois détaillées et constructives. Je remercie également M. François Clautiaux et M. Cyril Briand pour leur participation à ma soutenance de thèse et pour la pertinence de leurs questions et suggestions. Enfin, je souhaite adresser un remerciement particulier à M. Jean-Charles Billaut, avec qui j'ai fait mes premiers pas dans le domaine de la recherche opérationnelle sous la forme de plusieurs projets lors de mon parcours de formation d'ingénieur en informatique à Polytech Tours. C'est notamment grâce à lui que j'ai pu rencontrer ceux qui allaient devenir mes directeurs de thèse lors de mon stage de fin d'études d'ingénieur ; c'était donc un grand honneur pour moi qu'il accepte de présider mon jury. J'en profite pour remercier les enseignants-chercheurs de cette école pour la qualité de la formation dispensée ; c'était un plaisir de retrouver d'anciens professeurs en diverses occasions au cours de ma thèse.

En parallèle de mes activités de recherche, j'ai également eu l'opportunité d'enseigner dans deux contextes bien différents. D'un côté, j'ai enseigné en formation ingénieur et en master recherche à l'ISAE-SUPAERO sur des sujets liés à la recherche opérationnelle ; je remercie M. Alain Haït, M. Christophe Garion, Mme Stéphanie Roussel et M. Tanguy Pérennou de m'avoir accueilli avec bienveillance au sein de l'équipe enseignante. De l'autre côté, j'ai enseigné en licence informatique sur une thématique complètement différente (structures de données en langage C) ;

je remercie les intervenants de l'Université Paul Sabatier que j'ai eu l'occasion de rencontrer, notamment le responsable de cette unité d'enseignement, M. Mathias Paulin.

Je remercie tous les membres du Département d'Ingénierie des Systèmes Complexes de l'ISAE-SUPAERO pour tous les moments partagés au quotidien lors de ces trois années, en particulier M. Emmanuel Rachelson, Sébastien, Juan-José, Franco, Zoé, Ilia, Selma, Eyal, Anass, et bien évidemment mes chers collègues de bureau Luca et Erwan. Merci également aux membres de l'équipe Recherche Opérationnelle, Optimisation Combinatoire et Contraintes du LAAS-CNRS, à qui je rendais visite certes moins régulièrement mais avec autant de plaisir. Je pense notamment à Mme Sandra Ngueveu, pour son support aussi bien théorique que pratique, ainsi qu'à Margaux, Yun, Idir, Pierre, Mikael et Estèle. Je veux aussi saluer les membres du club de go de l'ONERA. Merci à tous pour les moments passés ensemble, à Toulouse et ailleurs, pendant les heures de travail comme en dehors.

Enfin, je tiens à remercier mes proches : amis de toujours (Antoine, Guillaume, Nicolas), amis de prépa (Audrey, Camille, Paolo), et aussi ma famille, en particulier ma mère, pour le soutien indéfectible qu'ils m'ont apporté tout au long de cette aventure.

Résumé

La structure de projet se retrouve dans de nombreux contextes de l'industrie et des services. Il s'agit de réaliser un ensemble d'activités pouvant être connectées par des liens logiques de séquence (antériorité), en faisant appel à des ressources disponibles en quantité limitée. L'objectif est la minimisation d'un critère généralement lié à la durée ou au coût du projet. La plupart des problèmes d'ordonnancement de projet dans la littérature considèrent une unité de temps commune pour la détermination des dates d'exécution des activités et pour l'évaluation instantanée du respect des capacités des ressources qu'elles utilisent. Or, s'il est souvent nécessaire en pratique d'obtenir un calendrier détaillé des plages d'exécution des activités, l'utilisation des ressources peut être évaluée sur un horizon plus agrégé, comme par exemple les quarts de travail des employés. Dans cette thèse, un nouveau modèle intégrant ces deux échelles de temps est présenté afin de définir le problème d'ordonnancement de projet avec agrégation périodique des contraintes de ressources (PARCPSP). Ce problème est étudié du point de vue de la théorie de la complexité et des propriétés structurelles sont établies, mettant notamment en évidence des différences majeures avec le problème classique d'ordonnancement de projet sous contraintes de ressources (RCPSP). De ces propriétés sont dérivées des formulations exactes basées sur la programmation linéaire en nombres entiers, comparées en termes de qualité de la relaxation linéaire. Par ailleurs, plusieurs heuristiques, telles que des algorithmes de liste, ou une méthode approchée basée sur une résolution itérative qui exploite différentes échelles de temps, sont proposées. Les résultats expérimentaux montrent l'intérêt de ces différentes méthodes et illustrent la difficulté du problème.

Mots-clés : planification ; ordonnancement ; projet ; optimisation combinatoire ; programmation linéaire en nombres entiers.

Abstract

The project structure arises in many fields of industry and services. It consists in performing a set of activities that may be linked by precedence relations, and use resources whose capacity is limited. The objective is to minimize a criterion usually linked to the duration or the cost of the project. Most of project scheduling problems in the literature assume that the same time scale should be used to determine activity start and completion dates and check resource constraints at each time. However, although it is often required in practice to build a precise schedule specifying the execution range of each activity, the resource usage can be evaluated on an aggregated basis, like worker shifts. In this thesis, a new model that enables the integration of these two time scales is presented in order to define the periodically aggregated resource-constrained project scheduling problem (PARCPSP). This problem is studied within the framework of complexity theory and several structural properties are established, highlighting major differences with the standard resource-constrained project scheduling problem (RCPSP). These properties allow deriving exact formulations based on integer linear programming, whose linear relaxations are compared. Moreover, several heuristics, such as schedule generations schemes, or an approached method based on a multi time scale iterative process, are proposed. Experimental results show the interest of these different methods and point out the intractability of the problem.

Keywords: planning; scheduling; project; combinatorial optimization; integer linear programming.

Table des matières

Table des figures

Liste des tableaux

Liste des Algorithmes

Introduction

De nos jours, toute entreprise doit gérer efficacement ses ressources (matérielles, humaines) afin de maintenir son niveau de compétitivité au plus haut. En effet, les impacts d'une gestion des ressources de qualité peuvent être extrêmement bénéfiques non seulement en termes de temps de réalisation des activités, mais aussi en termes de coût, permettant ainsi à l'entreprise de gagner en productivité. En dépit de ce constat, les fonctionnalités de planification proposées par les logiciels d'aide à la décision actuels quant à la gestion des ressources restent limitées à l'implémentation de méthodes simples, qui ne permettent pas d'englober tous les aspects de la gestion de projet. En particulier, ces outils ne permettent pas de prendre en compte différents niveaux de granularité temporelle pour la prise de décision concernant les divers éléments du projet. Si, par exemple, en termes de logistique, les dates de lancement des activités doivent être décidées avec une granularité assez fine sur l'horizon du projet, il n'est peut-être ni souhaitable ni nécessaire de planifier l'utilisation des ressources avec la même granularité. Par exemple, il est inutile, voire impossible, de concevoir un planning à long ou moyen terme dans lequel l'emploi du temps du personnel est détaillé pour chaque individu ; à ce stade, on préfèrera raisonner en termes de quantité de main d'œuvre disponible, selon les phases du projet. Notre objectif dans cette thèse est de proposer des modèles et des algorithmes d'ordonnancement et de planification susceptibles d'apporter une telle flexibilité en termes de granularité temporelle.

Dans la littérature consacrée à la gestion de projet, le problème le plus connu et le plus étudié est certainement le problème d'ordonnancement de ressources sous contraintes de ressources (Resource-Constrained Project Scheduling Problem, RCPSP), défini comme suit. Étant donnés un ensemble d'activités et de ressources, caractérisés par les durées des activités, leurs consommations en ressources, les capacités des ressources (nombres d'unités disponibles à chaque instant), ainsi que des relations d'antériorité entre les activités, l'objectif est de trouver un ordonnancement (c'est-à-dire choisir une date de début pour chaque activité) qui minimise la date de fin du projet (en supposant que celui commence à la date $t = 0$), en respectant deux types de contraintes. D'une part, les contraintes d'antériorité imposent qu'une activité ne peut pas être démarrée avant que ses prédécesseurs soient terminés. D'autre part, les contraintes de ressources imposent qu'à chaque instant, pour toute ressource, la somme des consommations des activités en cours d'exécution ne peut pas dépasser la capacité de la ressource.

Le RCPSP est l'un des problèmes combinatoires les plus difficiles à résoudre. Il existe encore à l'heure actuelle des instances de projets de petite taille (60 activités, 4 ressources) pour lesquelles l'optimalité n'a pu être prouvée pour aucune solution.

Si la version standard RCPSP reste un sujet de recherche actif encore au-

jourd'hui, de nombreuses variantes ont été proposées et également très étudiées dans la littérature. Par exemple, l'introduction de contraintes d'antériorité généralisées entre activités permettent de borner inférieurement et supérieurement le délai entre la fin d'une activité et le début d'un de ses successeurs. Dans le RCPSP multi-modes, il faut déterminer pour chaque activité non seulement sa date de début, mais aussi choisir un mode d'exécution (durée et consommations) parmi un ensemble de choix (fini et discret).

Le RCPSP est donc un modèle pertinent lorsqu'il est utilisé à un niveau de décision opérationnel (ordonnancement, à court terme). En revanche, ce modèle n'est pas assez flexible pour un niveau de décision tactique (planification, à moyen terme). En effet, à un niveau intermédiaire, l'estimation des paramètres du problème est soumise à des incertitudes. De plus, il est courant de procéder à une agrégation de l'information, par exemple en raisonnant non pas sur des tâches élémentaires mais des macro-tâches, ou encore en évaluant la consommation des tâches non pas précisément à chaque instant, mais en moyenne sur des périodes ; ainsi, les décisions prises au niveau tactique doivent permettre une certaine flexibilité, afin d'équilibrer la charge de travail dans chaque période et faciliter le calcul d'un ordonnancement réalisable au niveau opérationnel.

Dans cette thèse est proposé un modèle de gestion de projet situé entre les niveaux tactique et opérationnel, qui permet à la fois de définir précisément les dates de début et de fin des activités, notamment pour respecter les relations d'antériorité, tout en évaluant la consommation des tâches en ressources de manière agrégée sur un horizon discrétisé en périodes de même durée. La durée de ces périodes, paramétrable, devient ainsi un levier qui permet au décideur, pour un projet donné, de sélectionner le niveau d'agrégation le plus adapté au niveau de précision souhaité en termes de consommation des ressources.

La redéfinition des contraintes de ressources, évaluées non plus à chaque instant mais en moyenne dans chaque période, entraîne cependant une modification substantielle de la structure du problème, qui demande donc une étude théorique et pratique spécifique.

Ce manuscrit est structuré selon le plan suivant.

Tout d'abord, un état de l'art sur la programmation linéaire en nombres entiers pour l'ordonnancement de projet est proposé dans le chapitre 1, portant notamment sur des formulations indexées par le temps, ainsi que sur la notion d'ensemble admissible.

Les chapitres suivants traitent du problème principal étudié dans cette thèse, intitulé problème d'ordonnancement de projet sous contraintes de ressources avec agrégation périodique (PARCPSP).

Celui-ci est défini formellement dans le chapitre 2 et illustré sur quelques exemples. Ce chapitre comprend également une étude dans le cadre de la théorie de la complexité montrant que le PARCPSP est en général **NP**-difficile.

Une analyse des propriétés structurelles est menée dans le chapitre 3. À partir de concepts relatifs à l'altération de la faisabilité d'une solution par translation, le PARCPSP est comparé au RCPSP d'un point de vue théorique. L'impact de la durée des périodes sur la consommation moyenne des activités en ressources est également étudié.

Le chapitre 4 est dédié à des formulations exactes de programmation linéaire en nombres entiers pour le PARCPSP. Certaines propriétés du PARCPSP sont exploitées afin de renforcer ces modèles. Deux de ces modèles sont comparés d'un point de vue théorique en termes de qualité de leur relaxation linéaire.

Enfin, le chapitre 5 aborde des approches heuristiques pour le PARCPSP. Des algorithmes de liste sont adaptés à ce problème, et un algorithme itératif exploitant différentes échelles de temps est également proposé. Une analyse des résultats expérimentaux permet de comparer ces différentes méthodes de résolution.

Chapitre 1

Programmation linéaire en nombres entiers pour l'ordonnancement de projet

Ce chapitre présente un état de l'art succinct sur des approches de programmation linéaire en nombres entiers (PLNE) pour le problème d'ordonnancement de projet sous contraintes de ressources. Ces approches seront en effet utilisées pour résoudre le problème considéré dans cette thèse (voir chapitre 4). Pour introduire ce problème, nous présentons à la fin de ce chapitre différentes variantes qui considèrent une modélisation plus flexible des ressources.

1.1 RCPSP – Resource-Constrained Project Scheduling Problem

1.1.1 Définition du problème

Le problème d'ordonnancement de projet sous contraintes de ressources (RCPSP) est le problème de référence dans la littérature consacrée à la gestion de projet en tant que problème combinatoire. Un projet est défini par en ensemble de n activités à réaliser. Il existe des relations de séquence (antériorité) entre les activités : certaines activités doivent être terminées avant que d'autres puissent commencer. De plus, au cours de leur réalisation, les activités consomment une certaine quantité de différents types de ressources (m au total), disponibles en quantité limitée (capacité). On suppose que les ressources sont renouvelables : lorsqu'une activité se termine, l'ensemble des unités de ressources qu'elle consommait sont de nouveau disponibles et peuvent être réutilisées par d'autres activités. De plus, deux activités fictives, numérotées 0 et $n+1$, permettant de représenter respectivement le début et la fin du projet, sont ajoutées à l'ensemble des activités : leur durée est nulle (jalons), ainsi que leur consommation sur toutes les ressources. Les notations des données permettant de définir un projet sont répertoriées dans le tableau 1.1. On suppose que toutes les données numériques sont entières.

Le but est de trouver une solution qui minimise la durée du projet, tout en respectant les contraintes d'antériorité et de ressources. Une solution S définit, pour chaque activité $i \in \mathcal{A}$, une date de début d'exécution $S_i \in \mathbb{R}$. La préemption n'est

$\mathcal{A}$	Ensemble de n activités, plus deux activités fictives représentant le début et la fin du projet
$\mathcal{R}$	Ensemble de m ressources renouvelables
p_i	Durée d'exécution de l'activité $i \in \mathcal{A}$
b_k	Capacité de la ressource $k \in \mathcal{R}$
$r_{i,k}$	Consommation de l'activité $i \in \mathcal{A}$ sur la ressource $k \in \mathcal{R}$
E	$\subseteq \mathcal{A} \times \mathcal{A}$; ensemble de relations d'antériorité directes (graphe d'antériorité $= (\mathcal{A}, E)$)
E^*	Fermeture transitive de E, i.e., ensemble des relations d'antériorité directes et induites par transition

Tab. 1.1 : Données et notations pour le RCPSP

pas autorisée : une fois qu'une activité est démarrée, celle-ci ne peut pas être interrompue puis redémarrée. Ainsi, le RCPSP peut être défini de la manière suivante.

$$\text{Minimiser} \quad S_{n+1} - S_0 \tag{1.1}$$

$$S_{i_2} - S_{i_1} \geq p_{i_1} \qquad \forall (i_1, i_2) \in E \tag{1.2}$$

$$\sum_{i \in \mathcal{A}_t(S)} r_{i,k} \leq b_k \qquad \forall k \in \mathcal{R}\ ,\ \forall t \in \mathbb{R} \tag{1.3}$$

L'objectif (1.1) est de minimiser la durée du projet. Les contraintes d'antériorité (1.2) imposent, pour tout couple d'activités liées par une relation d'antériorité, que la fin d'exécution du prédécesseur se produise avant le début d'exécution du successeur. Les contraintes de ressources (1.3) imposent, sur chaque ressource, à tout instant, que la consommation totale à cet instant ne dépasse pas la capacité de la ressource. Ici, l'ensemble $\mathcal{A}_t(S)$ désigne l'ensemble des activités en cours d'exécution à la date t, en fonction du vecteur des dates de début S, défini par :

$$\mathcal{A}_t(S) = \{i \in \mathcal{A} \ :\ S_i \leq t < S_i + p_i\} \tag{1.4}$$

1.1.2 Conditions nécessaires et suffisantes d'existence de solutions

Dans le cas où les relations d'antériorité sont uniquement de type fin/début, les contraintes d'antériorité sont satisfaisables si, et seulement si, le graphe d'antériorité (pondéré par les durées des activités qui sont positives) est acyclique.

- Cette condition est nécessaire afin de pouvoir ordonnancer chaque activité.
- Cette condition est suffisante ; en effet, puisque le graphe d'antériorité est sans cycle, il représente un ordre partiel sur l'ensemble des activités, qui peut donc être étendu (de manière non unique *a priori*) en un ordre total. Ainsi, tout ordonnancement construit en exécutant les unes à la suite des autres les activités dans cet ordre (avec ou sans délai) respecte bien les contraintes d'antériorité.

Par ailleurs, les contraintes de ressources sont satisfaisables si, et seulement si, pour chaque activité, pour chaque ressource, la consommation de l'activité sur la

ressource ne dépasse pas sa capacité :

$$r_{i,k} \leq b_k \qquad \forall i \in \mathcal{A}\ ,\ \forall k \in \mathcal{R}$$

- Cette condition est nécessaire afin de pouvoir ordonnancer chaque activité.
- Cette condition est suffisante, car tout ordonnancement S tel que les fenêtres d'exécution des activités sont deux à deux disjointes respecte les contraintes de ressources (en effet, à tout instant $t \in \mathbb{R}$, l'ensemble $\mathcal{A}_t(S)$ contient au plus une activité).

Remarque. Tout ordonnancement construit à partir d'une extension de l'ordre partiel des activités, sans délai, respecte non seulement les contraintes d'antériorité, mais aussi les contraintes de ressources, puisque les fenêtres d'exécution des activités ne se chevauchent pas. La durée d'un tel ordonnancement est égale à la somme des durées des activités.

1.1.3 Règles de dominance et réductions de domaines

De par la définition des contraintes d'antériorité et de ressources, le fait de décaler une solution dans le temps n'affecte pas sa réalisabilité. Ainsi, il est possible d'imposer que le projet soit exécuté à partir de la date $S_0 = 0$; l'objectif est alors de minimiser la date de fin du projet, S_{n+1}.

Par ailleurs, puisque les données numériques sont entières, en particulier les durées des activités, on en déduit que les solutions entières (i.e. telles que toutes les activités commencent, et donc terminent, leur exécution à un instant entier) sont dominantes : s'il existe des solutions réalisables, alors il existe des solutions entières réalisables, et l'une d'entre elles (au moins) est optimale.

Il est ainsi possible de modifier la formulation précédente de la manière suivante :

$$\begin{aligned}
\text{Minimiser} \quad & S_{n+1} && && (1.1)\\
& S_{i_2} - S_{i_1} \geq p_{i_1} && \forall (i_1, i_2) \in E && (1.2)\\
& \sum_{i \in \mathcal{A}_t(S)} r_{i,k} \leq b_k && \forall k \in \mathbb{R}\ ,\ \forall t \in \mathbb{N} && (1.5)\\
& S_i \in \mathbb{N} && \forall i \in \mathbb{A} && (1.6)
\end{aligned}$$

Les contraintes de ressources (1.3) étaient vérifiées à tout instant continu ($t \in \mathbb{R}$) ; ici, les contraintes de ressources (1.5) sont vérifiées seulement à des instants entiers ($t \in \mathbb{N}$), puisque la consommation sur la période de durée unitaire $[t, t+1[$ n'évolue pas : en effet, le domaine des dates de début des activités est restreint à l'ensemble des entiers positifs (contraintes 1.6).

Par ailleurs, il est possible de déterminer un horizon de planification, i.e. une borne supérieure de la durée du projet. Comme cela a été remarqué précédemment, une borne triviale est la somme des durées des activités. De plus, en calculant, dans le graphe d'antériorité pondéré par les durées des activités, les plus longs chemins depuis l'activité fictive de début et vers l'activité fictive de fin, il est possible de resserrer les domaines des dates de début des activités, et également obtenir une borne inférieure de la durée du projet. Les notations relatives à ces paramètres temporels supplémentaires sont répertoriées dans le tableau 1.2.

Il est à noter que d'autres techniques de filtrage permettent de réduire le domaine des dates de début (raisonnement énergétique, règles *not first/not last*).

h	Horizon de planification (borne supérieure de la durée du projet)
$\mathcal{T}$	$= \{0, \dots, h\}$; ensemble des instants entiers considérés, de $t = 0$ jusqu'à l'horizon de planification h
ES_i	Date de début au plus tôt *(earliest start)* de l'activité $i \in \mathcal{A}$ Cette valeur ne dépend pas de l'horizon de planification h.
LS_i	Date de début au plus tard *(latest start)* de l'activité $i \in \mathcal{A}$ Cette valeur dépend de l'horizon de planification h.

Tab. 1.2 : Paramètres temporels supplémentaires pour le RCPSP

1.2 Formulations indexées par les périodes

Cette section est dédiée à la présentation d'une famille de modélisations exactes pour le RCPSP, basées sur la programmation linéaire en nombres entiers. Plus précisément, ces formulations sont dites indexées par les périodes. En effet, dans ces formulations, la date de début d'une activité n'est pas représentée par une unique variable entière S_i, mais par un ensemble de variables binaires indexées par le temps, qui permettent d'encoder la valeur de S_i. Cet encodage est particulièrement utile afin de modéliser les contraintes de ressources.

1.2.1 Trois types de variables binaires

Dans la littérature, trois types de variables binaires ont été proposés. Les encodages correspondant sont illustrés pour chaque type dans la figure 1.1 page 8.

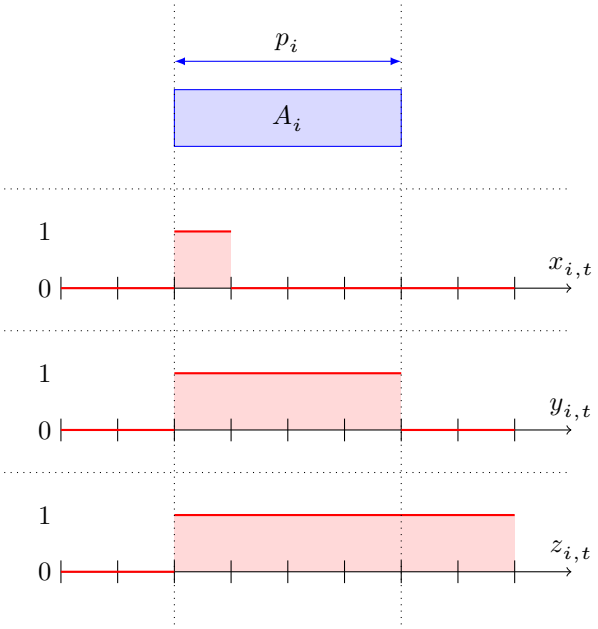


Fig. 1.1 : Trois types de variables binaires permettant de modéliser l'exécution d'une activité

1. Variables *pulse* $x_{i,t}$
   Pour chaque activité $i \in \mathcal{A}$, parmi l'ensemble des variables $x_{i,t}$ ($t \in \mathcal{T}$), une et une seule d'entre elle vaut 1 tandis que toutes les autres valent 0.

$$x_{i,t} = 1 \quad \Leftrightarrow \quad S_i = t$$

2. Variables *on/off* $y_{i,t}$
   Pour chaque activité $i \in \mathcal{A}$, pour tout $t \in \mathcal{T}$, $y_{i,t}$ vaut 1 si et seulement si l'activité i est en cours d'exécution à la date t, i.e., $i \in \mathcal{A}_t(S)$.

$$y_{i,t} = 1 \quad \Leftrightarrow \quad S_i \leq t < S_i + p_i$$

3. Variables *step* $z_{i,t}$
   Pour chaque activité $i \in \mathcal{A}$, parmi l'ensemble des variables $z_{i,t}$ ($t \in \mathcal{T}$), toutes celles situées avant S_i (i.e. $t < S_i$) valent 0, tandis que toutes celles situées après (i.e. $t \geq S_i$) valent 1, formant ainsi un front montant en S_i.

$$z_{i,t} = 1 \quad \Leftrightarrow \quad S_i \leq t$$

D'après la signification de chaque type de variables, pour toute activité $i \in \mathcal{A}$, les valeurs des variables en dehors de la fenêtre $[ES_i, LS_i]$ sont connues (y compris, par convention, pour toute date t en dehors de l'horizon de planification) :

- Si $t < ES_i$ (y compris $t < 0$), alors $x_{i,t} = 0$, $y_{i,t} = 0$ et $z_{i,t} = 0$.
- Si $t > LS_i$ (y compris $t > h$), alors $x_{i,t} = 0$, $y_{i,t} = 0$ et $z_{i,t} = 1$.

Comme cela a été montré par Sousa (1989), ces trois manières d'encoder la date de début d'une activité sont en fait équivalentes : en effet, puisqu'il existe des bijections linéaires permettant de passer de chaque type de variable à chaque autre, les relaxations linéaires des modèles présentés dans la suite de ce chapitre (obtenues en remplaçant les variables binaires par des variables continues bornées entre 0 et 1) sont des descriptions différentes d'un même polytope (pour plus de détails, voir Artigues 2017). Ces six changements de variables sont détaillés dans la proposition ci-après.

Proposition 1.1. *Il existe des bijections linéaires permettant de passer d'un type de variables à l'autre.*

Démonstration. De par le sens attribué à chaque type de variables, il est aisé d'exprimer les variables *pulse* ($x_{i,t}$) en fonction des variables *step* ($z_{i,t}$).

$$\begin{aligned} x_{i,t} = 1 \quad &\Leftrightarrow \quad S_i = t \\ &\Leftrightarrow \quad t - 1 < S_i \leq t \\ &\Leftrightarrow \quad (S_i > t - 1) \wedge (S_i \leq t) \\ &\Leftrightarrow \quad \big(z_{i,t-1} = 0\big) \wedge \big(z_{i,t} = 1\big) \\ &\Leftrightarrow \quad z_{i,t} - z_{i,t-1} = 1 \end{aligned}$$

D'où :

$$x_{i,t} = z_{i,t} - z_{i,t-1} \tag{1.7}$$

La transformation inverse est la suivante :

$$z_{i,t} = \sum_{t'=0}^{t} x_{i,t'} \tag{1.8}$$

De même, il est aisé d'exprimer les variables *on/off* ($y_{i,t}$) en fonction des variables *step* ($z_{i,t}$).

$$\begin{aligned} y_{i,t} = 1 \quad &\Leftrightarrow \quad t - p_i < S_i \leq t \\ &\Leftrightarrow \quad (S_i > t - p_i) \wedge (S_i \leq t) \\ &\Leftrightarrow \quad \left(z_{i,t-p_i} = 0\right) \wedge \left(z_{i,t} = 1\right) \\ &\Leftrightarrow \quad z_{i,t} - z_{i,t-p_i} = 1 \end{aligned}$$

D'où :

$$y_{i,t} = z_{i,t} - z_{i,t-p_i} \tag{1.9}$$

La transformation inverse est la suivante :

$$z_{i,t} = \sum_{t' \in t - p_i \mathbb{N}} y_{i,t'} \tag{1.10}$$

où : $t - p_i\mathbb{N} = \{t - p_i\,\alpha \mid \alpha \in \mathbb{N}\}$

En composant ces relations, il est possible d'exprimer les variables *on/off* ($y_{i,t}$) en fonction des variables *pulse* ($x_{i,t}$) et vice versa.

$$y_{i,t} = \sum_{t'=t-p_i+1}^{t} x_{i,t'} \tag{1.11}$$

$$x_{i,t} = \sum_{t' \in t - p_i \mathbb{N}} y_{i,t'} - \sum_{t' \in t - 1 - p_i \mathbb{N}} y_{i,t'} \tag{1.12}$$

□

Les sections suivantes, 1.2.2 à 1.2.4, présentent les formulations PLNE du RCPSP basées sur ces trois types de variables binaires. Dans la suite de ce chapitre, on choisira d'appliquer les conventions suivantes pour les activités fictives.

- L'activité fictive de début de projet est ignorée. Ainsi, tout se passe comme ci celle-ci avait une durée nulle et des consommations nulles sur toutes les ressources.
- L'activité fictive de fin de projet a une durée non pas nulle mais unitaire, et des consommations nulles sur toutes les ressources. Ainsi, cette activité est la seule qui peut être exécutée dans la dernière période de l'horizon $[h, h+1[$; l'exécution des activités non fictives du projet peut donc se faire uniquement sur l'intervalle $[0, h[$.

1.2.2 Modèle basé sur des variables pulse

Le modèle basé sur les variables *pulse* $x_{i,t}$, initialement introduit par Pritsker et al. (1969), permet d'encoder la date de début S_i de la manière suivante :

$$x_{i,t} = 1 \quad \Leftrightarrow \quad S_i = t$$

Ainsi, il est possible d'exprimer S_i comme une combinaison linéaire des variables binaires $x_{i,t}$.

$$S_i = \sum_{t \in \mathcal{T}} t\, x_{i,t}$$

Par ailleurs, pour déterminer si l'activité i est en cours d'exécution à l'instant t, il suffit de tester si l'activité i a démarré à l'un des instants $t' \in \{t - p_i + 1, \cdots, t\}$.

$$i \in \mathcal{A}_t(S) \quad \Leftrightarrow \quad \sum_{t'=t-p_i+1}^{t} x_{i,t'} = 1$$

Le modèle basé sur les variables *pulse* s'écrit donc de la manière suivante.

$$\text{Minimiser} \quad \sum_{t \in \mathcal{T}} t\, x_{n+1,t} \tag{1.13}$$

$$\sum_{t \in \mathcal{T}} t\, x_{i_2,t} - \sum_{t \in \mathcal{T}} t\, x_{i_1,t} \geq p_{i_1} \qquad \forall (i_1, i_2) \in E \tag{1.14}$$

$$\sum_{i \in \mathcal{A}} \sum_{t'=t-p_i+1}^{t} r_{i,k}\, x_{i,t'} \leq b_k \qquad \forall k \in \mathcal{R}\ ,\ \forall t \in \mathcal{T} \tag{1.15}$$

$$\sum_{t \in \mathcal{T}} x_{i,t} = 1 \qquad \forall i \in \mathcal{A} \tag{1.16}$$

$$x_{i,t} \in \{0, 1\} \qquad \forall i \in \mathcal{A}\ ,\ \forall t \in \mathcal{T} \tag{1.17}$$

L'objectif (1.13) est de minimiser la date de fin du projet, sous contraintes d'antériorité (1.14) et de ressources (1.15). Les contraintes (1.16) permettent de modéliser le comportement *pulse* des variables $x_{i,t}$ binaires (contraintes 1.17).

Désagrégation des contraintes d'antériorité Les contraintes (1.14) sont appelées contraintes d'antériorité agrégées : chaque relation d'antériorité (arc du graphe d'antériorité) est modélisée par une unique contrainte. Il est possible de remplacer chacune de ces contraintes agrégées par un ensemble de contraintes indexées par le temps, dites désagrégées. La désagrégation est basée sur la traduction en contraintes linéaires des implications suivantes :

$$S_{i_2} \leq t \quad \Rightarrow \quad S_{i_1} \leq t - p_{i_1} \qquad \forall (i_1, i_2) \in E$$

Ainsi, les contraintes d'antériorité agrégées (1.14) peuvent être remplacées par les contraintes d'antériorité désagrégées (1.18) suivantes ; le modèle ainsi obtenu est celui introduit par Christofides et al. (1987).

$$\sum_{t'=0}^{t-p_{i_1}} x_{i_1,t'} - \sum_{t'=0}^{t} x_{i_2,t'} \geq 0 \qquad \forall (i_1, i_2) \in E\ ,\ \forall t \in \mathcal{T} \tag{1.18}$$

Remarque. Pour une relation d'antériorité donnée $(i_1, i_2) \in E$, en sommant les contraintes désagrégées (1.18) sur l'horizon $\mathcal{T}$, on obtient la contrainte agrégée (1.14) correspondante. La désagrégation des contraintes d'antériorité permet en fait de renforcer strictement la relaxation linéaire du modèle.

1.2.3 Modèle basé sur des variables step

Le modèle basé sur les variables *step* $z_{i,t}$, initialement introduit par Pritsker et Watters (1968), permet d'encoder la date de début S_i de la manière suivante :

$$z_{i,t} = 1 \quad \Leftrightarrow \quad S_i \leq t$$

Ainsi, il est possible d'exprimer S_i comme une combinaison linéaire des variables binaires $z_{i,t}$.

$$S_i = h - \sum_{t \in \mathcal{T}} z_{i,t}$$

Par ailleurs, pour déterminer si l'activité i est en cours d'exécution à l'instant t, il suffit de tester si le démarrage de l'activité i a eu lieu à la t ou après, mais pas à la date $t - p_i$ ou avant.

$$i \in \mathcal{A}_t(S) \quad \Leftrightarrow \quad z_{i,t} - z_{i,t-p_i} = 1$$

Le modèle basé sur les variables *step* s'écrit donc de la manière suivante.

$$\text{Minimiser} \quad h - \sum_{t \in \mathcal{T}} z_{n+1,t} \tag{1.19}$$

$$\sum_{t \in \mathcal{T}} (z_{i_1,t} - z_{i_2,t}) \geq p_{i_1} \qquad \forall (i_1, i_2) \in E \tag{1.20}$$

$$\sum_{i \in \mathcal{A}} r_{i,k} \left(z_{i,t} - z_{i,t-p_i} \right) \leq b_k \qquad \forall k \in \mathcal{R} \ , \ \forall t \in \mathcal{T} \tag{1.21}$$

$$z_{i,h} = 1 \qquad \forall i \in \mathcal{A} \tag{1.22}$$

$$z_{i,t} - z_{i,t-1} \geq 0 \qquad \forall i \in \mathcal{A} \tag{1.23}$$

$$z_{i,t} \in \{0, 1\} \qquad \forall i \in \mathcal{A} \ , \ \forall t \in \mathcal{T} \tag{1.24}$$

L'objectif (1.19) est de minimiser la date de fin du projet, sous contraintes d'antériorité (1.20) et de ressources (1.21). Les contraintes (1.22) assurent que chaque activité est bien réalisée au cours de l'exécution du projet. Les contraintes (1.23) permettent de modéliser le comportement *step* des variables $z_{i,t}$ binaires (contraintes 1.24).

Désagrégation des contraintes d'antériorité Comme précédemment, les contraintes d'antériorité agrégées (1.20) peuvent être remplacées par les contraintes d'antériorité désagrégées (1.25) suivantes ; le modèle ainsi obtenu est celui introduit par Souza et Wolsey (1997).

$$z_{i_1,t-p_{i_1}} - z_{i_2,t} \geq 0 \qquad \forall (i_1, i_2) \in E \ , \ \forall t \in \mathcal{T} \tag{1.25}$$

1.2.4 Modèle basé sur des variables on/off

Le modèle basé sur les variables *on/off* $y_{i,t}$, initialement introduit par Sousa (1989), permet d'encoder la date de début S_i de la manière suivante :

$$y_{i,t} = 1 \quad \Leftrightarrow \quad S_i \leq t < S_i + p_i$$

Ainsi, il est possible d'exprimer S_i comme une combinaison linéaire des variables binaires $y_{i,t}$.

$$S_i = h - \sum_{t \in \mathcal{T}} \left\lceil \frac{h - t}{p_i} \right\rceil y_{i,t}$$

Par ailleurs, les variables $y_{i,t}$, par définition, permettent de déterminer directement si l'activité i est en cours d'exécution à l'instant t.

$$i \in \mathcal{A}_t(S) \quad \Leftrightarrow \quad y_{i,t} = 1$$

Le modèle basé sur les variables *on/off* s'écrit donc de la manière suivante.

$$\text{Minimiser} \quad \sum_{t\in\mathcal{T}} t\, y_{n+1,t} \tag{1.26}$$

$$\sum_{t\in\mathcal{T}} \left\lceil \frac{h-t}{p_{i_1}} \right\rceil y_{i_1,t} - \sum_{t\in\mathcal{T}} \left\lceil \frac{h-t}{p_{i_2}} \right\rceil y_{i_2,t} \geq p_{i_1} \qquad \forall (i_1,i_2)\in E \tag{1.27}$$

$$\sum_{i\in\mathcal{A}} r_{i,k}\, y_{i,t} \leq b_k \qquad \forall k\in\mathcal{R}\ ,\ \forall t\in\mathcal{T} \tag{1.28}$$

$$\sum_{t\in h-p_i\mathbb{N}} y_{i,t} = 1 \qquad \forall i\in\mathcal{A} \tag{1.29}$$

$$\sum_{t'\in t-p_i\mathbb{N}} y_{i,t'} - \sum_{t'\in t-1-p_i\mathbb{N}} y_{i,t'} \geq 0 \qquad \forall i\in\mathcal{A} \tag{1.30}$$

$$y_{i,t}\in\{0,1\} \qquad \forall i\in\mathcal{A}\ ,\ \forall t\in\mathcal{T} \tag{1.31}$$

L'objectif (1.26) est de minimiser la date de fin du projet, sous contraintes d'antériorité (1.27) et de ressources (1.28). Les contraintes (1.29) assurent que chaque activité est bien réalisée au cours de l'exécution du projet. Les contraintes (1.30) permettent de modéliser le comportement *on/off* des variables $y_{i,t}$ binaires (contraintes 1.31).

Remarque. L'expression de la fonction objectif (1.26) a été simplifiée de la manière suivante. Rappelons que, par convention, la durée de l'activité fictive de fin p_{n+1} est égale à 1. Par conséquent, d'après la contrainte (1.29) :

$$\sum_{t\in h-p_{n+1}\mathbb{N}} y_{n+1,t} = \sum_{t\in\mathcal{T}} y_{n+1,t} = 1$$

D'où :

$$\begin{aligned} S_{n+1} &= h - \sum_{t\in\mathcal{T}} \left\lceil \frac{h-t}{p_{n+1}} \right\rceil y_{n+1,t} \\ &= h - \sum_{t\in\mathcal{T}} (h-t)\, y_{n+1,t} \\ &= h - h\underbrace{\sum_{t\in\mathcal{T}} y_{n+1,t}}_{=1} + \sum_{t\in\mathcal{T}} t\, y_{n+1,t} \\ &= \sum_{t\in\mathcal{T}} t\, y_{n+1,t} \end{aligned}$$

Désagrégation des contraintes d'antériorité Comme précédemment, les contraintes d'antériorité agrégées (1.27) peuvent être remplacées par les contraintes d'antériorité désagrégées (1.32) suivantes ; le modèle ainsi obtenu a également été étudié par Sousa (1989).

$$\sum_{t'\in t-p_{i_1}-p_{i_1}\mathbb{N}} y_{i_1,t'} - \sum_{t'\in t-p_{i_2}\mathbb{N}} y_{i_2,t'} \geq 0 \qquad \forall (i_1,i_2)\in E\ ,\ \forall t\in\mathcal{T} \tag{1.32}$$

1.3 Formulations étendues basées sur la notion d'ensemble d'activités admissible

Cette section est consacrée à des formulations alternatives aux modèles précédents. Ces formulations, dites étendues, sont basées sur la notion d'ensemble

d'activités admissible, qui permet d'isoler les contraintes de ressources. Dans la littérature, ce type de formulations est connu pour donner des bornes inférieures de très bonne qualité.

Tout d'abord, la notion d'ensemble d'activités admissible est définie. Ensuite, une formulation issue de la littérature utilisant ce concept est présentée. Enfin, les formulations indexées par le temps étudiées dans la section précédente sont décomposées selon ce même principe, faisant ainsi apparaître un problème maître spécifique au type de variables utilisé (*pulse*, *step*, *on-off*), et un sous-problème identique, pouvant être intégrés dans un schéma de résolution de type *branch-and-price* (combinant *branch-and-bound* et génération de colonnes).

Dans la littérature, cette formulation a été essentiellement présentée uniquement sur les variables de type *pulse*. Un état de l'art sur ces formulations pour les trois types de variables a été présenté en conférence (Morin, Artigues, Haït et Ngueveu 2017).

1.3.1 Ensemble d'activités admissible

Définition 1.1 (Admissibilité d'un ensemble d'activités)**.** Un (sous-)ensemble d'activités est admissible si, et seulement si, toutes les activités qu'il contient peuvent être exécutées simultanément tout en respectant les contraintes d'antériorité et de ressources.

L'ensemble des notations relatives aux ensembles d'activités (admissibles ou non) sont répertoriées dans le tableau 1.3.

$\mathcal{L}$	Ensemble des ensembles d'activités $\ell \subseteq \mathcal{A}$ admissibles
$a_{i,\ell}$	Donnée binaire égale à 1 si l'activité $i \in \mathcal{A}$ appartient à l'ensemble $\ell \subseteq \mathcal{A}$ (i.e. $i \in \ell$), 0 sinon
ES_ℓ	$= \max_{i \in \ell} (ES_i)$ Date à partir de laquelle l'ensemble ℓ peut être en cours d'exécution
LS_ℓ	$= \min_{i \in \ell} (LS_i)$ Date jusqu'à laquelle l'ensemble ℓ peut être en cours d'exécution
$\mathcal{L}_t$	Ensemble des ensembles d'activités admissibles $\ell \in \mathcal{L}$ pouvant être en cours d'exécution à la date t, i.e. tels que $ES_\ell \leq t \leq LS_\ell$

Tab. 1.3 : Notations associées aux ensembles d'activités

Les ensembles d'activités admissibles sont donc les solutions du programme linéaire en nombres entiers suivant, dans lequel la variable binaire α_i indique, pour chaque activité $i \in \mathcal{A}$, si celle-ci appartient à l'ensemble ($\alpha_i = 1$) ou non ($\alpha_i = 0$) ; ainsi, la solution renvoyée par cette formulation est l'ensemble admissible $\ell \in \mathcal{L}$ tel que $a_{i,\ell} = \alpha_i$ pour toute activité $i \in \mathcal{A}$.

$$\alpha_{i_1} + \alpha_{i_2} \leq 1 \qquad \forall (i_1, i_2) \in E^* \tag{1.33}$$

$$\sum_{i \in \mathcal{A}} r_{i,k}\, \alpha_i \leq b_k \qquad \forall k \in \mathcal{R} \tag{1.34}$$

$$\alpha_i \in \{0, 1\} \qquad \forall i \in \mathcal{A} \tag{1.35}$$

Ce modèle n'a pas de fonction objectif, seulement des contraintes (problème de satisfaction). Les contraintes (1.33) assurent que les relations d'antériorité sont bien respectées, tandis que les contraintes (1.34) assurent que les capacités des ressources ne sont pas dépassées. Enfin, les variables α_i sont binaires (contraintes 1.35).

Dans les modèles présentés dans la suite de cette section, les contraintes de ressources ne sont plus exprimées par des contraintes de sac à dos dans lesquelles une expression binaire indique, pour chaque activité $i \in \mathcal{A}$, si elle est en cours d'exécution à un instant donné $t \in \mathcal{T}$ (i.e. $i \in \mathcal{A}_t(S)$). Ces modèles utilisent des variables binaires $Y_{\ell,t}$ qui prennent la valeur 1 si l'ensemble d'activités en cours d'exécution à $t \in \mathcal{T}$ est $\ell \in \mathcal{L}_t$, 0 sinon. Ces variables doivent être liées aux variables binaires associées aux activités.

Dans la littérature

La première formulation basée sur la notion d'ensembles admissibles à été proposée par Mingozzi et al. (1998) qui présentent le modèle suivant.

$$\text{Minimiser} \quad \sum_{t \in \mathcal{T}} t\, x_{n+1,t} \tag{1.36}$$

$$\sum_{t \in \mathcal{T}} t\, x_{i_2,t} - \sum_{t \in \mathcal{T}} t\, x_{i_1,t} \geq p_{i_1} \qquad \forall (i_1, i_2) \in E \tag{1.37}$$

$$\sum_{t \in \mathcal{T}} x_{i,t} = 1 \qquad \forall i \in \mathcal{A} \tag{1.38}$$

$$\sum_{\ell \in \mathcal{L}} Y_{\ell,t} \leq 1 \qquad \forall t \in \mathcal{T} \tag{1.39}$$

$$\sum_{\ell \in \mathcal{L}} \sum_{t \in \mathcal{T}} a_{i,\ell}\, Y_{\ell,t} = p_i \qquad \forall i \in \mathcal{A} \tag{1.40}$$

$$x_{i,t} \geq \sum_{\ell \in \mathcal{L}} a_{i,\ell}\, Y_{\ell,t} - \sum_{\ell \in \mathcal{L}} a_{i,\ell}\, Y_{\ell,t-1} \qquad \forall t \in \mathcal{T} \tag{1.41}$$

$$x_{i,t} \in \{0,1\} \qquad \forall i \in \mathcal{A}\ ,\ \forall t \in \mathcal{T} \tag{1.42}$$

$$Y_{\ell,t} \in \{0,1\} \qquad \forall \ell \in \mathcal{L}\ ,\ \forall t \in \mathcal{T} \tag{1.43}$$

L'objectif (1.36) est de minimiser la date de fin du projet. Les contraintes (1.37) modélisent les relations d'antériorité. Les contraintes (1.38) modélisent le comportement *pulse* des variables $x_{i,t}$. Les contraintes (1.39) indiquent que, pour $t \in \mathcal{T}$ fixé, au plus une seule variable $Y_{\ell,t}$ ($\ell \in \mathcal{L}$)* peut prendre la valeur 1. Dans ce cas, toutes les activités dans l'ensemble ℓ sont en cours d'exécution dans la période unitaire $[t, t+1[$. Sinon, toutes ces variables valent 0 : aucune activité n'est en cours d'exécution dans cette période (i.e. l'ensemble d'activités en cours d'exécution dans cette période est l'ensemble vide). Les contraintes (1.40), combinées aux contraintes (1.41), forcent la synchronisation des variables $x_{i,t}$ (liées aux activités) et des variables $Y_{\ell,t}$ (liées aux ensembles d'activités) tout en assurant que la durée d'exécution de chaque activité est bien respectée. Enfin, toutes les variables sont binaires (contraintes 1.42 et 1.43).

Dans la suite de ce chapitre, nous présentons les modèles obtenus en décomposant de manière similaire les formulations indexées par le temps introduites dans

*L'ensemble $\mathcal{L}$ pourrait ici être remplacé par $\mathcal{L}_t$.

la section précédente. La liaison entre les variables binaires liées aux activités ($x_{i,t}$, $y_{i,t}$ ou $z_{i,t}$) et les variables indiquant l'ensemble en cours d'exécution à un instant donné ($Y_{\ell,t}$) est faite de manière différente, en traduisant l'équivalence suivante en contraintes linéaires.

$$Y_{\ell,t} = 1 \quad \Leftrightarrow \quad \Big(\forall i \in \mathcal{A} \quad (i \in \mathcal{A}_t(S)) \;\Leftrightarrow\; (a_{i,\ell} = 1) \Big)$$

1.3.2 Problème maître

Nous proposons ici un schéma de décomposition pour les trois formulations indexées par le temps présentées précédemment. Seule la séparation du problème écrit avec des variables *pulse* $x_{i,t}$ est détaillée ici ; en utilisant les bijections linéaires identifiées dans la preuve de la proposition 1.1 page 9, il est possible d'obtenir les deux autres modèles équivalents qui utilisent soit les variables *on/off* $y_{i,t}$, soit les variables *step* $z_{i,t}$.

$$\text{Minimiser} \quad \sum_{t \in \mathcal{T}} t\, x_{n+1,t} \tag{1.13}$$

$$\sum_{t \in \mathcal{T}} t\, x_{i_2,t} - \sum_{t \in \mathcal{T}} t\, x_{i_1,t} \geq p_{i_1} \qquad \forall (i_1, i_2) \in E \tag{1.14}$$

$$\sum_{t \in \mathcal{A}} x_{i,t} = 1 \qquad \forall i \in \mathcal{T} \tag{1.16}$$

$$-\sum_{\ell \in \mathcal{L}_i} Y_{\ell,t} \geq -1 \qquad \forall t \in \mathcal{T} \tag{1.44}$$

$$\sum_{t'=t-p_i+1}^{t} x_{i,t'} - \sum_{\ell \in \mathcal{L}_t} a_{i,\ell}\, Y_{\ell,t} = 0 \qquad \forall i \in \mathcal{A}\ ,\ \forall t \in \mathcal{T} \tag{1.45}$$

$$x_{i,t} \in \{0,1\} \qquad \forall i \in \mathcal{A}\ ,\ \forall t \in \mathcal{T} \tag{1.17}$$

$$Y_{\ell,t} \geq 0 \qquad \forall t \in \mathcal{T}\ ,\ \forall \ell \in \mathcal{L}_t \tag{1.46}$$

L'objectif (1.13) est de minimiser la date de fin du projet. Les contraintes (1.14) permettent de modéliser les relations d'antériorité. Les contraintes (1.16) permettent de modéliser le comportement *pulse* des variables $x_{i,t}$. Les contraintes (1.44) imposent qu'une seule variable $Y_{\ell,t}$ représente l'ensemble en cours d'exécution à chaque instant ; on note V_t les variables duales associées à ces contraintes. Les contraintes (1.45) permettent de synchroniser les variables $x_{i,t}$ aux variables $Y_{\ell,t}$; on note $W_{i,t}$ les variables duales associées à ces contraintes. Enfin, les contraintes (1.17) et (1.46) définissent le domaine des variables.

Dans ce modèle, inspiré de celui proposé par Ngueveu et al. (2016) pour un autre problème d'ordonnancement, les variables $Y_{\ell,t}$ sont implémentées comme des variables continues ; l'ensemble des contraintes assure qu'elles sont (implicitement) binaires. Finalement, la validité du modèle défini par les équations (1.44) à (1.46) et (1.13) à (1.17) découle de la proposition suivante.

Proposition 1.2. *À chaque instant $t^* \in \mathcal{T}$, l'une ou bien l'autre des configurations suivantes se produit.*

1. $\forall \ell \in \mathcal{L}_{t^*} \quad Y_{\ell,t^*} = 0$
   Aucune activité n'est en cours d'exécution à l'instant t^*.

2. $\exists!\, \ell^* \in \mathcal{L}_{t^*} \quad \big(Y_{\ell^*,t^*} = 1\big) \wedge \big(\forall \ell \in \mathcal{L}_{t^*} \setminus \{\ell^*\} \quad Y_{\ell,t^*} = 0\big)$
L'ensemble des activités en cours d'exécution à l'instant t^* est ℓ^*, identifié par une seule et unique variable/colonne (celle qui vaut 1, toutes les autres s'annulant).

Démonstration. D'après les contraintes (1.46), $Y_{\ell,t^*} \geq 0$ pour tout $\ell \in \mathcal{L}_{t^*}$; de plus, d'après la contrainte (1.44) à l'instant t^*, $Y_{\ell,t^*} \leq 1$ pour tout $\ell \in \mathcal{L}_{t^*}$.

1. Si : $\forall \ell \in \mathcal{L}_{t^*} \quad Y_{\ell,t^*} = 0$
   Dans cette configuration, d'après la contrainte (1.45) pour toute activité $i \in \mathcal{A}$ à l'instant t^* :
   $\sum_{t=t^*-p_i+1}^{t^*} x_{i,t} = 0$
   Autrement dit, aucune activité n'est en cours d'exécution à l'instant t^*.
2. Sinon : $\exists \ell^* \in \mathcal{L}_{t^*} \quad 0 < Y_{\ell^*,t^*} \leq 1$
   Cette configuration peut être encore partitionnée en deux cas.
   (a) Si : $Y_{\ell^*,t^*} = 1$
   Dans ce cas, d'après la contrainte (1.44) à l'instant t^*, $Y_{\ell,t^*} = 0$ pour tout $\ell \in \mathcal{L}_{t^*} \setminus \{\ell^*\}$.
   Donc, d'après la contrainte (1.45) pour toute activité $i \in \mathcal{A}$ à l'instant t^* :
   $\sum_{t=t^*-p_i+1}^{t^*} x_{i,t} = a_{i,\ell^*}$
   Autrement dit, l'ensemble des activités en cours d'exécution à t^* est exactement ℓ^*, repéré par l'unique variable qui vaut 1, toutes les autres étant nulles.
   (b) Sinon : $0 < Y_{\ell^*,t^*} < 1$
   Montrons par l'absurde que ce cas ne peut pas se produire.
      i. Si : $\forall \ell \in \mathcal{L}_{t^*} \setminus \{\ell^*\} \quad Y_{\ell,t^*} = 0$
      Puisque $\ell^* \neq \emptyset$, il existe donc une activité $i^* \in \ell^*$ (i.e. $a_{i^*,\ell^*} = 1$).
      Donc, d'après la contrainte (1.45) pour l'activité i^* à l'instant t^* :
      $\sum_{t=t^*-p_{i^*}+1}^{t^*} x_{i^*,t} = Y_{\ell^*,t^*}$
      On remarque que le membre de gauche est entier (et même binaire) tandis que le membre de droite est compris strictement entre 0 et 1 : contradiction.
      ii. Sinon : $\exists \ell' \in \mathcal{L}_{t^*} \setminus \{\ell^*\} \quad 0 < Y_{\ell',t^*} < 1$
      (borne supérieure stricte obtenue d'après la contrainte (1.44) à l'instant t^*)
      Puisque $\ell' \neq \ell^*$, l'une ou l'autre des affirmations suivantes (symétriques) est vraie.
      - $\exists i^* \in \ell^* \quad i^* \notin \ell' \qquad$ (i.e. $a_{i^*,\ell^*} = 1$ et $a_{i^*,\ell'} = 0$)
      - $\exists i' \in \ell' \quad i' \notin \ell^* \qquad$ (i.e. $a_{i',\ell'} = 1$ et $a_{i',\ell^*} = 0$)

      Sans perte de généralité (quitte à échanger ℓ^* et ℓ'), supposons que la première des deux affirmations est vraie.
      D'après la contrainte (1.45) pour l'activité i^* à l'instant t^* :
      $\sum_{t=t^*-p_{i^*}+1}^{t^*} x_{i^*,t} = \sum_{\ell \in \mathcal{L}_{t^*} \setminus \{\ell^*,\ell'\}} a_{i^*,\ell}\, Y_{\ell,t^*} + 1\, Y_{\ell^*,t^*} + 0\, Y_{\ell',t^*}$
      Par conséquent :
      - $\sum_{t=t^*-p_{i^*}+1}^{t^*} x_{i^*,t} \geq Y_{\ell^*,t^*} > 0$
      - $\sum_{t=t^*-p_{i^*}+1}^{t^*} x_{i^*,t} < \sum_{\ell \in \mathcal{L}_{t^*}} Y_{\ell,t^*} \leq 1$

      Le membre de gauche, entier (et même binaire), est compris stricte-

ment entre 0 et 1 : contradiction. □

1.3.3 Sous-problème

L'ensemble $\mathcal{L}$ des ensembles d'activités admissibles est de taille exponentielle en fonction du nombre d'activités dans le pire des cas. Afin d'éviter une énumération exhaustive de cet ensemble, une approche de type *branch-and-price* permettant de générer les variables/colonnes $Y_{\ell,t}$ au cours du processus de résolution peut être mise en place. La contrainte duale associée à la variable $Y_{\ell,t}$ s'écrit de la manière suivante :

$$-V_t - \sum_{i \in \ell} W_{i,t} \leq 0 \qquad \forall t \in \mathcal{T}\ ,\ \forall \ell \in \mathcal{L}_t \tag{1.47}$$

D'une part, il s'agit bien d'une contrainte de type $\leq$, puisque la variable primale correspondante $Y_{\ell,t}$ est positive. D'autre part, le second membre est bien nul, puisque les variables $Y_{\ell,t}$ n'apparaissent pas dans les fonctions objectifs des problèmes maîtres.

Étant donné un instant $t \in \mathcal{T}$, le modèle PLNE suivant permet de trouver un ensemble admissible $\ell \in \mathcal{L}_t$ tel que la variable $Y_{\ell,t}$ est améliorante, i.e. viole la contrainte duale (1.47), ou permet de prouver qu'un tel ensemble n'existe pas.

$$\text{Minimiser} \quad V_t + \sum_i W_{i,t}\,\alpha_i \tag{1.48}$$

$$\alpha_{i_1} + \alpha_{i_2} \leq 1 \qquad \forall (i_1, i_2) \in E^* \tag{1.49}$$

$$\sum_i r_{i,k}\,\alpha_i \leq b_k \qquad \forall k \in \mathcal{R} \tag{1.50}$$

$$\alpha_i \in \{0,1\} \qquad \forall i \in \mathcal{A}\ :\ ES_i \leq t \leq LS_i \tag{1.51}$$

$$\alpha_i = 0 \qquad \forall i \in \mathcal{A}\ :\ (t < ES_i) \vee (LS_i < t) \tag{1.52}$$

L'objectif (1.48) est de minimiser le coût réduit. Les contraintes (1.49) à (1.51) sont identiques aux contraintes (1.33) à (1.35) introduites en début de section, permettant de caractériser l'ensemble des ensembles d'activités admissibles. Enfin, les contraintes (1.52) permettent de restreindre l'espace de recherche à $\mathcal{L}_t$ (sans elles, l'espace de recherche est égal à $\mathcal{L}$).

Cette thèse est consacrée à l'étude d'une variante du RCPSP pour laquelle les contraintes de ressources sont agrégées. Afin de modéliser ce problème, des formulations PLNE reprenant les formulations présentées dans la section 1.2 seront proposées dans le chapitre 4. Le passage à des formulations étendues pour cette variante reste une perspective à approfondir.

1.4 Extensions du RCPSP dans la littérature

Le RCPSP est le problème standard de référence dans la littérature. Les variantes de ce problème sont également très nombreuses. Cette section est dédiée à la présentation de quelques unes de ces variantes, qui permettent de gérer les ressources de manière plus flexible, tout en conservant les relations d'antériorité

entre les activités. Par conséquent, ces problèmes sont soit adaptés à un haut niveau de décision (stratégique, voire tactique), soit à un niveau de décision plus bas (ordonnancement) lorsqu'ils permettent de modéliser précisément la manière dont les ressources sont gérées. C'est le cas par exemple dans le problème considéré par Artigues et al. (2009), où deux échelles de temps distinctes sont considérées simultanément : l'une, détaillée, permet de déterminer précisément les dates de début et fin des tâches, tandis que l'autre, plus agrégée, permet de déterminer la quantité de travail réalisée par chaque opérateur sur un quart de travail.

Le Rough Cut Capacity Planning (RCCP), introduit par Hans (2001), est défini de la manière suivante. L'horizon de planification est constitué de périodes de durée *a priori* non unitaire. Les événements de début et de fin d'exécution des activités peuvent avoir lieu à n'importe quel instant (continu) : ils ne correspondent donc pas nécessairement aux bornes des périodes. La contrainte sur l'utilisation des ressources n'est pas instantanée, mais périodique : en moyenne, la quantité de ressources consommées par les activités ne doit pas dépasser la capacité de la ressource. De plus, la quantité de ressources allouée à une activité donnée dans une période donnée, appelée intensité, est également une variable de décision (qui dépend de l'activité, de la ressource, et de la période). Hans propose une méthodologie de type Branch-and-Price pour ce problème : le problème maître gère l'affectation des ressources aux activités, tandis que le sous-problème renvoie des colonnes indiquant, pour une période donnée, quelles sont les activités qui peuvent être exécutées dans cette période. Le modèle proposé autorise l'exécution de deux activités prédécesseur/successeur dans une même période ; dans ce cas, la contrainte d'antériorité correspondante peut être violée par la solution renvoyée.

Pour assurer le respect des relations d'antériorité, il est possible d'ajouter une contrainte supplémentaire : si une activité termine son exécution dans une certaine période, alors l'exécution de ses successeurs ne peut commencer qu'à partir de la période suivante. Cela mène à la définition d'un autre problème, intitulé RCPSP avec intensité variable (RCPSVP), introduit par Kis (2005, 2006), qui propose plusieurs programmes linéaires mixtes, ainsi qu'une étude polyédrale à partir de laquelle sont dérivées des inégalités valides liées aux relations d'antériorité, incorporées dans une approche de type Branch-and-Cut.

Il est néanmoins possible de modéliser de manière exacte les contraintes d'antériorité, sans pour autant surcontraindre le problème. Pour y parvenir, Haït et Baydoun (2012) ont proposé une formulation linéaire en nombres entiers originale, basée sur une représentation temporelle mixte. Plus précisément, leur modèle utilise à la fois des variables continues représentant les dates de début et de fin d'exécution des activités, et des variables indexées par les périodes (discrétisation agrégée). L'avantage de ce type de modélisation est la facilité d'expression des contraintes en utilisant les variables appropriées en fonction des contraintes (variables temporelles continues pour les contraintes d'antériorité, variables indexées par périodes agrégées pour les contraintes de ressources). En revanche, une difficulté supplémentaire apparaît : le modèle doit assurer une cohésion temporelle globale, i.e. faire en sorte que les valeurs prises par les différents types de variables temporelles soient cohérentes.

Par ailleurs, dans la littérature, d'autres types de ressources ont été considérés. Par exemple, les ressources non renouvelables, contrairement aux ressources renouvelables, ne peuvent pas être réutilisées après avoir été consommées : à la

fin de l'exécution du projet, la quantité totale consommée ne doit pas dépasser la quantité initialement disponible. Ce type de ressources devient pertinent lorsqu'il existe plusieurs manières différentes d'exécuter une activité (souvent appelées modes d'exécution dans la littérature). Typiquement, la durée d'une activité peut être réduite si on lui affecte plus de ressources.

Böttcher et al. (1999) ont proposé une généralisation de ces deux types de ressources, en introduisant le concept de ressource partiellement renouvelable, définissant ainsi une variante du problème d'ordonnancement de projet sous contraintes de ressources classique, nommée RCPSP/π. Une ressource partiellement renouvelable k est caractérisée par un couple (b_k, π_k), où $b_k \in \mathbb{N}$ est le nombre d'unités disponibles (capacité), et $\pi_k \subseteq \mathcal{T}$ est l'union d'un sous-ensemble de périodes unitaires, consécutives ou non, durant lesquelles cette ressource est disponible. Ce type de ressources permet donc de modéliser finement la variation de la disponibilité des ressources au cours du temps (par exemple, différence entre les semaines et les week-ends). De plus, les ressources renouvelables et non-renouvelables sont des cas particuliers de ressources partiellement renouvelables :

- Une ressource non-renouvelable k de capacité b_k est équivalente à une unique ressource partiellement renouvelable de même capacité, dont la disponibilité recouvre l'ensemble des période unitaires : $(b_k, \mathcal{T})$
- Une ressource renouvelable k de capacité b_k est équivalente à un ensemble de ressources partiellement renouvelables de même capacité, dont les disponibilités sont des périodes unitaires, deux à deux disjointes, recouvrant l'horizon : $\{(b_k, \{t\}) \ : \ t \in \mathcal{T}\}$.

Ce problème est défini de manière purement discrète : les données numériques sont entières, et les dates de début/fin d'activité doivent être entières. Contrairement aux problèmes précédents (RCPSP, RCCP, RCPSVP), pour lesquels il existe des conditions nécessaires et suffisantes sur l'existence de solutions réalisables, déterminer s'il existe des solutions pour une instance donnée du RCPSP/π est **NP**-difficile.

Cette thèse est consacrée à l'étude d'une variante située à la croisée des problèmes cités ci-avant, intitulée problème d'ordonnancement de projets avec agrégation périodique des contraintes de ressources (periodically aggregated resource constrained project scheduling problem, ou PARCPSP). Ce problème peut être défini de manière équivalente à partir des différents problèmes vus précédemment. Comme pour le RCCP et le RCPSVP, l'évaluation de la consommation des ressources est également agrégée par périodes ; en revanche, les durées et les consommations des activités sont connues et fixes (ce ne sont plus des variables de décision qui dépendent de l'intensité ; cela revient à dire que l'intensité est fixée à 100%). Le PARCPSP peut également être vu comme une relaxation d'un cas particulier du RCPSP/π lorsque l'horizon de planification est découpé uniformément (les périodes sont toutes de même durée) : la contrainte d'intégrité des dates de début/fin d'activités est retirée (passage en temps continu).

Chapitre 2

PARCPSP : définition, complexité

Dans ce chapitre, nous présentons le problème d'ordonnancement de projet avec agrégation périodique des ressources (PARCPSP). Nous donnons une formulation conceptuelle du problème et une condition nécessaire et suffisant d'existence d'une solution réalisable. Nous illustrons au moyen d'exemples quelques propriétés de base de ce problème, et notamment des différences essentielles avec le RCPSP (qui admet le PARCPSP comme relaxation), à savoir la non dominance des solutions entières et la non-invariance de la propriété de faisabilité d'une solution par translation. Enfin, une étude de complexité de différents cas particuliers du problème montrent qu'en général le PARCPSP est **NP**-difficile au sens fort.

2.1 Définition du PARCPSP : Periodically Aggregated Resource-Constrained Project Scheduling Problem

2.1.1 Données

Les données d'entrée du PARCPSP peuvent être séparées en deux parties indépendantes : un projet (mêmes données que pour le RCPSP) et une subdivision temporelle de l'horizon de planification utile pour l'évaluation des contraintes de ressources agrégées.

Projet

Un projet est défini par un ensemble $\mathcal{A}$ composé de $n \in \mathbb{N}$ activités et un ensemble $\mathcal{R}$ composé de $m \in \mathbb{N}$ ressources. Chaque activité $i \in \mathcal{A}$ est définie par sa durée d'exécution notée p_i (entier strictement positif). Chaque ressource $k \in \mathcal{R}$ est définie par sa capacité (quantité maximale disponible) notée b_k (entier strictement positif). À chaque instant de son exécution, l'activité $i \in \mathcal{A}$ consomme $r_{i,k}$ unités de la ressource $k \in \mathcal{R}$ (entier positif). Les activités sont soumises à des relations d'antériorité, représentées par un ensemble $E \subset \mathcal{A} \times \mathcal{A}$ de couples prédécesseur/successeur (i_1, i_2).

De plus, deux activités fictives additionnelles sont considérées : l'activité 0 qui représente le début du projet (prédécesseur de toute tâche non fictive), et l'activité $n+1$ qui représente la fin du projet (successeur de toute tâche non fictive). Ces

deux activités fictives ont une durée d'exécution nulle et une consommation nulle sur toutes les ressources.

La convention de représentation d'un projet sous forme graphique est indiquée dans la figure 2.1 (projet avec $n = 3$ activités ; l'activité 1 précède l'activité 2, tandis que l'activité 3 n'est soumise à aucune relation d'antériorité).

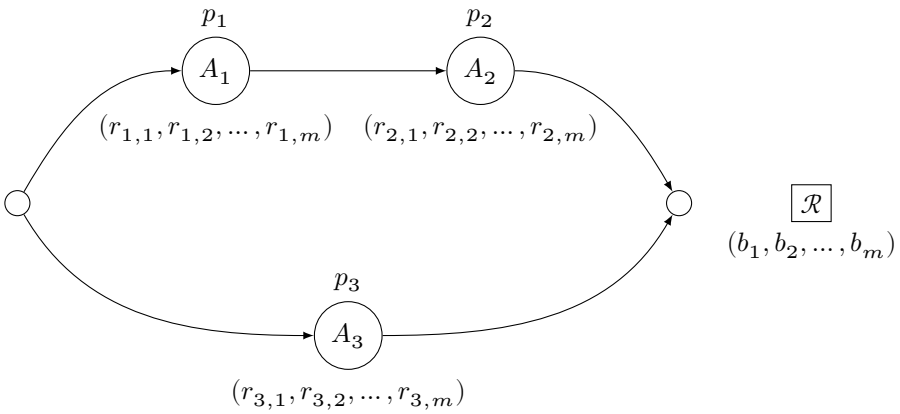


Fig. 2.1 : Représentation graphique d'un projet

On note $\mathcal{X}$ l'ensemble des projets.

Subdivision temporelle de l'horizon de planification

L'horizon de planification ($\mathbb{R}$) est subdivisé uniformément en périodes de même longueur $\Delta \in \mathbb{R}_+^*$. Plus précisément, la période $\ell \in \mathbb{Z}$ correspond à l'intervalle $[(\ell-1)\Delta, \ell\Delta]$; ainsi, la période 1 correspond à l'intervalle $[0, \Delta]$ (cf. figure 2.2).

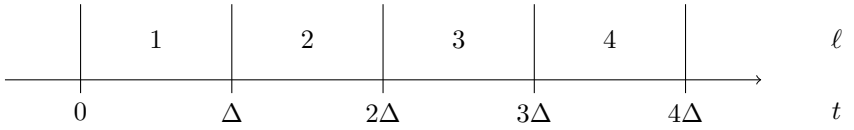


Fig. 2.2 : Subdivision temporelle uniforme de l'horizon de planification

2.1.2 Représentation d'une solution en termes de dates et de consommation de ressources

Représentation minimale Une solution du PARCPSP est représentée par un vecteur $S \in \mathbb{R}^n$, où S_i désigne la date de début d'exécution de l'activité $i \in \mathcal{A}$.

Bien qu'une solution puisse être représentée de manière minimale par les dates de début d'exécution des activités, il est utile de pouvoir caractériser une solution en termes d'intersections des fenêtres d'exécution des activités avec les périodes de l'horizon temporel, afin d'être en mesure de calculer la consommation agrégée de chaque activité sur chaque période.

Évaluation de la durée d'exécution dans une période donnée en fonction de la date de début d'exécution Afin de définir la contrainte sur la ressource

$k \in \mathcal{R}$ agrégée dans la période $\ell \in \mathbb{Z}$, il est nécessaire de pouvoir exprimer la consommation moyenne de toute activité $i \in \mathcal{A}$ dans la période ℓ.

Soit $D_{i,\ell} : \mathbb{R} \to [0, \Delta]$ la fonction qui, à un instant $t \in \mathbb{R}$, associe la longueur de l'intersection de deux intervalles (cf. figure 2.3) :

- $[t, t + p_i]$, i.e. la fenêtre d'exécution de l'activité i si celle-ci commence à la date t ;
- $[(\ell - 1)\Delta, \ell\Delta]$, i.e. la période ℓ.

$$D_{i,\ell}(t) = \max\left(\ 0\ ,\ \min\left(\ell\Delta, t + p_i\right) - \max\left((\ell - 1)\Delta, t\right)\ \right) \tag{2.1}$$

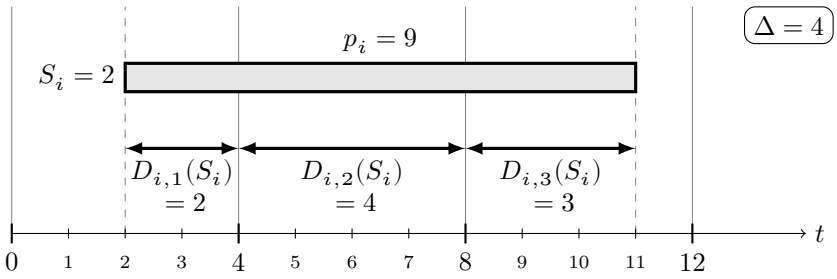


Fig. 2.3 : Évaluation de la durée d'exécution dans une période

Consommation moyenne d'une activité dans une période Puisque l'activité i consomme $r_{i,k}$ unités de la ressource k à chaque instant de son exécution, la consommation moyenne de l'activité i dans la période ℓ (de durée Δ) pour une solution $S \in \mathbb{R}^n$ est donc égale à $r_{i,k}\frac{D_{i,\ell}(S_i)}{\Delta}$ (cf. figure 2.4 dans laquelle la consommation agrégée du PARCPSP est illustrée en regard de la consommation instantanée du RCPSP).

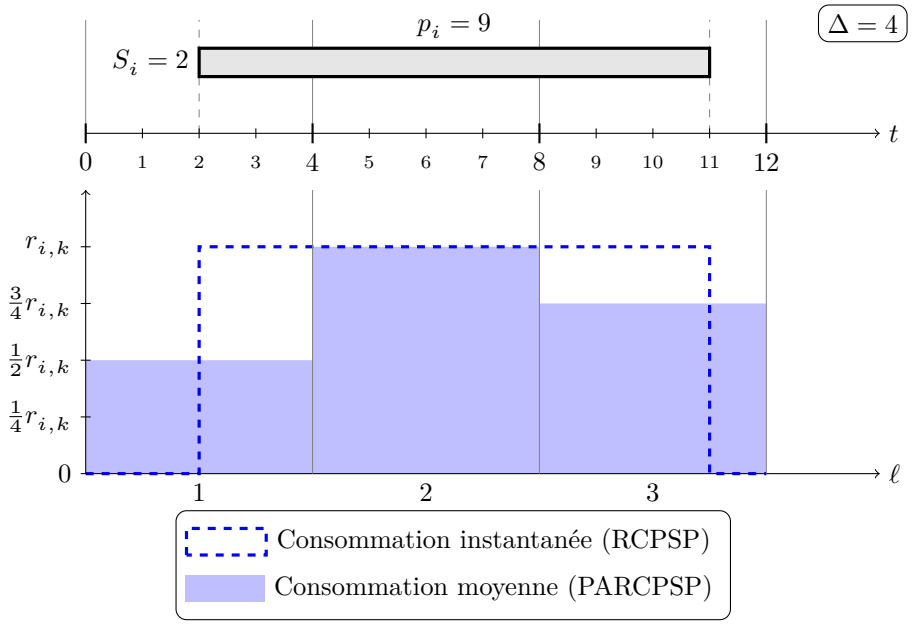


Fig. 2.4 : Calcul de la consommation moyenne d'une activité par période

2.1.3 Formulation abstraite

Le PARCPSP peut être formulé de la manière suivante :

$$\text{Minimiser :} \quad S_{n+1} - S_0 \tag{2.2}$$

$$S_{i_2} - S_{i_1} \geq p_{i_1} \qquad \forall (i_1, i_2) \in E \tag{2.3}$$

$$\sum_{i \in \mathcal{A}} r_{i,k} \frac{D_{i,\ell}(S_i)}{\Delta} \leq b_k \qquad \forall k \in \mathcal{R}\ ,\ \forall \ell \in \mathbb{Z} \tag{2.4}$$

Fonction objectif Comme pour le RCPSP, l'objectif du PARCPSP consiste à minimiser la durée du projet. Puisque l'horizon temporel est agrégé en périodes, plusieurs mesures de la durée du projet peuvent être envisagées *a priori*.

- Différence entre la plus petite date de début d'exécution et la plus grande
- Temps écoulé depuis la date $t = 0$ jusqu'à la plus grande date de fin
- Nombre de périodes sur lesquelles le projet est exécuté (différence entre les indices des périodes au cours desquelles le projet commence/se termine plus 1).

Dans cette thèse, la durée du projet est définie conformément à la première mesure proposée, c'est-à-dire, $S_{n+1} - S_0$ (2.2). La deuxième mesure (S_{n+1}), appelée *makespan* dans la littérature et souvent notée $C_{\max}$, est également considérée, uniquement dans la section 2.3 dédiée à l'analyse de la complexité du PARCPSP.

Contraintes Comme pour le RCPSP, des contraintes d'antériorité (2.3) indiquent que, pour tout couple prédécesseur/successeur $(i_1, i_2) \in E$, l'exécution de i_1 doit être achevée avant que celle de i_2 puisse commencer.

De plus, des contraintes de ressource (2.4) sont également considérées. Par rapport au RCPSP, ces contraintes sont relâchées. Plus précisément, la consommation instantanée n'est pas bornée ici ; seule la consommation moyenne sur des périodes de durée Δ ne peut pas dépasser la capacité de la ressource.

Ainsi, le PARCPSP est une relaxation du RCPSP ; ce résultat sera formalisé plus tard (cf. proposition 3.4).

2.1.4 Dominance des solutions commençant dans la première période

Puisque l'horizon temporel est subdivisé uniformément, il est possible, sans perte de généralité, de faire commencer le projet dans la première période $[0, \Delta]$, i.e. de supposer que $0 \leq S_0 \leq \Delta$.

Pour une solution donnée, cette transformation peut être interprétée de deux manières : soit comme un décalage de cette solution dans le temps par une quantité multiple de la durée des périodes Δ, soit comme un décalage de la numérotation des périodes. Dans un cas comme dans l'autre, la faisabilité de la solution est préservée, et la durée du projet n'est pas modifiée.

2.1.5 Existence de solutions réalisables

Contraintes d'antériorité Comme pour le RCPSP, les contraintes d'antériorité sont satisfaisables si et seulement si le graphe d'antériorité est acyclique.

Contraintes de ressources agrégées par période Pour le RCPSP, il est nécessaire et suffisant que la consommation de chaque activité sur chaque ressource ne dépasse pas la capacité, i.e. $r_{i,k} \leq b_k$, pour toute activité $i \in \mathcal{A}$, pour toute ressource $k \in \mathcal{R}$. Pour le PARCPSP, cette condition reste suffisante mais elle n'est plus nécessaire.

Proposition 2.1. *Les contraintes de ressources agrégées par période sont satisfaisables si et seulement si :*

$$\forall i \in \mathcal{A} \quad \forall k \in \mathcal{R} \qquad r_{i,k} \leq b_k \max\left(1, \frac{2\Delta}{p_i}\right)$$

Démonstration. Il existe un ordonnancement qui respecte les contraintes de ressources agrégées par période si et seulement si chaque activité peut être ordonnancée seule tout en respectant ces contraintes : autrement dit, il doit exister, pour chaque activité $i \in \mathcal{A}$, une date de début $t_i \in \mathbb{R}$ telle que l'ordonnancement dans lequel l'activité i est exécutée seule à partir de la date t_i (fenêtre d'exécution $[t_i, t_i + p_i]$) est réalisable. Puisque l'horizon temporel est subdivisé uniformément en périodes de durée $\Delta \in \mathbb{R}_+^*$, on peut supposer, sans perte de généralité, que $0 \leq t_i < \Delta$.

Soit $i \in \mathcal{A}$. On suppose qu'il existe un tel t_i pour cette activité. Ainsi, pour toute ressource $k \in \mathbb{R}$, dans toute période $\ell \in \mathbb{Z}$:

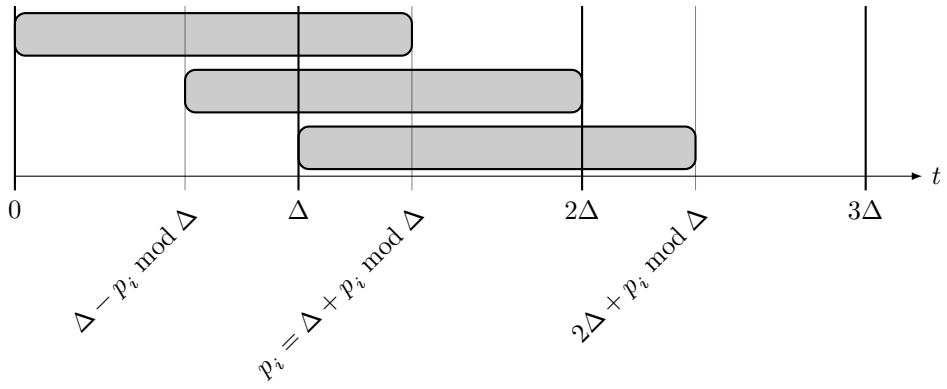
$$r_{i,k}\, D_{i,\ell}(t_i) \leq b_k\, \Delta$$

Cet ensemble de contraintes est satisfaisable si et seulement si, pour toute ressource $k \in \mathbb{R}$:

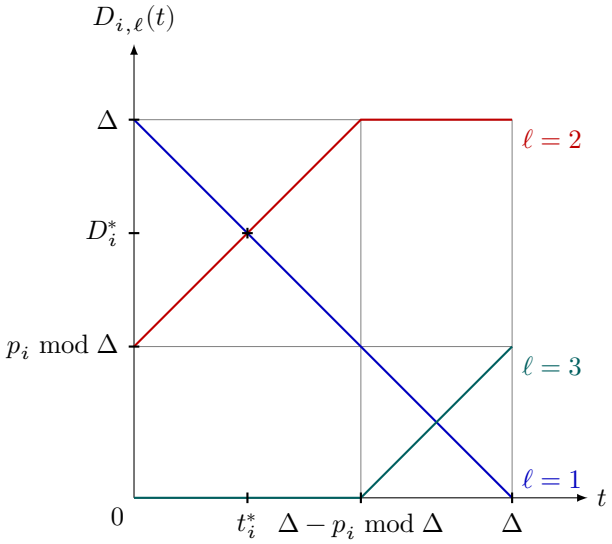
$$r_{i,k} \max_{\ell \in \mathbb{Z}} D_{i,\ell}(t_i) \leq b_k\, \Delta$$

On obtient ainsi une condition toujours suffisante, mais non nécessaire en général, sauf si t_i minimise le terme $\max_{\ell \in \mathbb{Z}} D_{i,\ell}(t_i)$. Soit $D_i^* = \min_{0 \leq t < \Delta} \max_{\ell \in \mathbb{Z}} D_{i,\ell}(t)$ la valeur minimale de ce maximum, atteinte pour un certain t_i^* ($0 \leq t_i^* < \Delta$).

- $p_i \geq 2\Delta$
  Dans ce cas, quelle que soit la date de début d'exécution $t \in \mathbb{R}$ choisie, il existe toujours une période $\ell \in \mathbb{Z}$ incluse dans la fenêtre d'exécution, i.e. $[(\ell-1)\Delta, \ell\Delta] \subseteq [t, t+p_i]$.
  Ainsi, dans cette période : $D_{i,\ell}(t) = \Delta$
  D'où : $D_i^* = \Delta$
- $\Delta \leq p_i < 2\Delta$
  Une étude des variations des fonctions $D_{i,\ell}$ qui ne sont pas nulles en tout point $t \in [0, \Delta]$ est proposée dans la figure 2.5 page 26.
  On peut remarquer que $D_i^* = \frac{p_i \bmod \Delta + \Delta}{2} = \frac{p_i}{2}$; cette configuration est atteinte pour $t_i^* = \frac{\Delta - p_i \bmod \Delta}{2} = \Delta - \frac{p_i}{2}$.
  En particulier, $D_{i,1}(t_i^*) = D_{i,2}(t_i^*) = \frac{p_i}{2}$, tandis que $D_{i,\ell}(t_i^*) = 0$ dans toutes les autres périodes ($\ell \notin \{1, 2\}$) : la fenêtre d'exécution de l'activité i est séparée en deux parties de même durée sur deux périodes consécutives (cf. figure 2.7 page 28).
- $0 < p_i < \Delta$
  Une étude des variations des fonctions $D_{i,\ell}$ qui ne sont pas nulles en tout point $t \in [0, \Delta]$ est proposée dans la figure 2.6 page 27.
  On peut remarquer que $D_i^* = \frac{p_i}{2}$; cette configuration est atteinte pour $t_i^* = \frac{(\Delta - p_i) + \Delta}{2} = \Delta - \frac{p_i}{2}$.

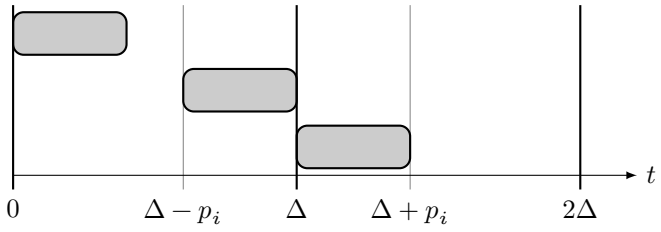


(a) Cas limites

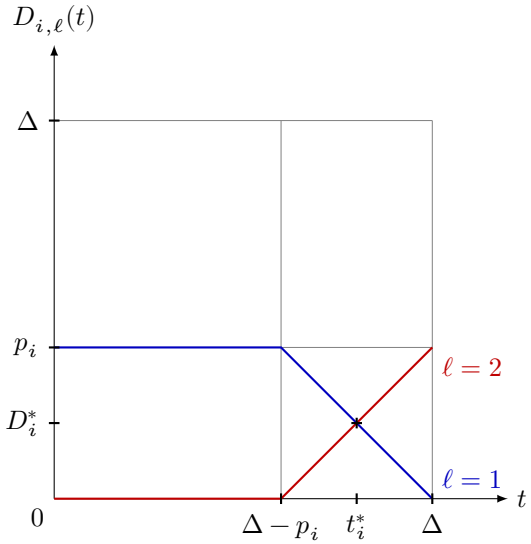


(b) Représentation graphique de $D_{i,\ell}(t)$

Fig. 2.5 : Existence de solutions réalisables par rapport aux contraintes de ressources agrégées par période : cas $\Delta \leq p_i < 2\Delta$



(a) Cas limites



(b) Représentation graphique de $D_{i,\ell}(t)$

Fig. 2.6 : Existence de solutions réalisables par rapport aux contraintes de ressources agrégées par période : cas $0 < p_i < \Delta$

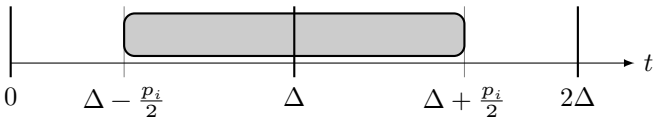


Fig. 2.7 : Existence de solutions réalisables par rapport aux contraintes de ressources agrégées par période : configuration pour laquelle D_i^* est atteint ($0 < p_i < 2\Delta$)

En particulier, $D_{i,1}(t_i^*) = D_{i,2}(t_i^*) = \frac{p_i}{2}$, tandis que $D_{i,\ell}(t_i^*) = 0$ dans toutes les autres périodes ($\ell \notin \{1, 2\}$) : il s'agit en fait de la même configuration que pour le cas précédent.

Par conséquent : $D_i^* = \min\left(\Delta, \frac{p_i}{2}\right)$

D'où : $r_{i,k}\, D_i^* \leq b_k\, \Delta \quad \Leftrightarrow \quad r_{i,k} \leq b_k\, \frac{\Delta}{D_i^*} = b_k\, \max\left(1, \frac{2\Delta}{p_i}\right)$ □

2.2 Exemples

Exemple 1 Considérons le projet représenté dans la figure 2.8a page 29, défini par deux activités identiques de durée unitaire, sans relation d'antériorité, utilisant chacune une unité d'une même ressource de capacité unitaire au cours de leur exécution.

Dans le cas du RCPSP, c'est-à-dire avec des contraintes de ressources "classiques" (à chaque instant), la ressource est disjonctive : il est impossible que les fenêtres d'exécution des deux activités se chevauchent, même partiellement. Ainsi, dans toute solution optimale, les deux activités sont exécutées l'une après l'autre, sans temps mort : la durée minimale du projet est donc égale à 2.

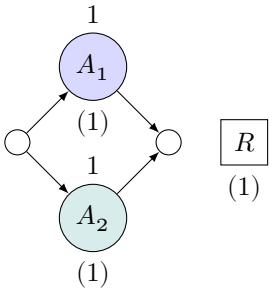
Intéressons-nous désormais au cas du PARCPSP défini avec des périodes de durée unitaire ($\Delta = 1$), et, plus particulièrement, à la faisabilité de solutions telles que les activités sont exécutées simultanément (la durée du projet est alors égale à 1).

- Si l'exécution des deux activités commence la date $t = 0 = S_1 = S_2$ (voir figure 2.8b), elle termine alors à la date $t = 1$. Ces deux instants correspondent aux bornes de la première période $[0; 1]$. Dans cette période, la consommation moyenne de chaque activité est égale à 1. Par conséquent, la consommation moyenne totale est égale à 2 ; elle est nulle dans toutes les autres périodes. Puisque la capacité de la ressource est unitaire, cette solution n'est donc pas réalisable.

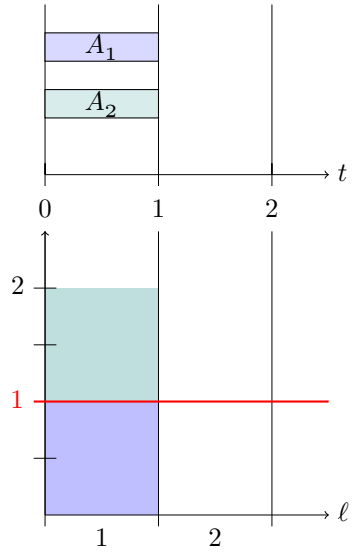
  On peut remarquer que toute solution entière S' telle que $S'_1 = S'_2$ peut être obtenue en décalant cette solution par un facteur entier, c'est-à-dire un multiple de la durée des périodes (puisque $\Delta = 1$) : ces solutions sont donc toutes irréalisables.

  Par conséquent, si une solution entière S' est réalisable, alors $S'_1 \neq S'_2$, d'où :

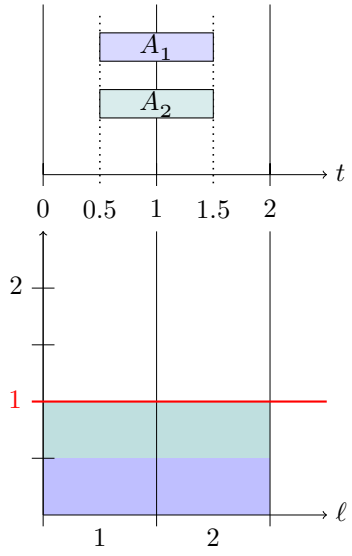
$$\begin{aligned} S'_{n+1} - S'_0 &= \max\left(S'_1 + p_1, S'_2 + p_2\right) - \min\left(S'_1, S'_2\right) \\ &= \max\left(S'_1, S'_2\right) + 1 - \min\left(S'_1, S'_2\right) \\ &= |S'_2 - S'_1| + 1 \geq 2 \end{aligned}$$



(a) Projet



(b) Solution non réalisable ($\Delta = 1$)



(c) Solution réalisable ($\Delta = 1$)

Fig. 2.8 : Exemple #1

- Si l'exécution des deux activités commence la date $t = 0,5 = S_1 = S_2$ (voir figure 2.8b), elle termine alors à la date $t = 1,5$. Deux périodes sont donc intersectées par les fenêtres d'exécution : $[0;1]$ et $[1;2]$. Dans ces deux périodes, la consommation moyenne de chaque activité est égale à $0,5$. Par conséquent, dans ces deux périodes, la consommation moyenne totale est égale à 1 ; elle est nulle dans toutes les autres périodes. Cette fois, il s'agit d'une solution réalisable, et même optimale (puisque $S_{n+1} - S_0 = 1,5 - 0,5 = 1 = p_1 = p_2$).

Ce premier exemple permet de mettre en avant certaines différences structurelles importantes entre le RCPSP (contraintes de ressources à chaque instant) et le PARCPSP (contraintes de ressources agrégées par période), qui seront étudiées plus en détail dans le chapitre 3.

- Les solutions entières sont dominantes pour le RCPSP ; elles ne le sont plus pour le PARCPSP.
- Pour le RCPSP, décaler une solution dans le temps n'affecte pas sa faisabilité ; ainsi, on peut supposer que l'exécution du projet commence à la date 0, ce qui implique que la durée du projet est égale à la plus grande date de fin d'exécution (makespan, noté $C_{\max}$).
  En revanche, pour le PARCPSP, décaler une solution dans le temps peut affecter sa faisabilité. De plus, il n'existe pas toujours de solution optimale commençant à la date 0. La durée du projet n'est donc plus égale au makespan. Néanmoins, avec une subdivision temporelle uniforme, il est possible de supposer que l'exécution du projet commence dans la première période $[0;\Delta]$: en effet, un décalage temporel $\tau \in \mathbb{R}$ dont la valeur est un multiple de la durée des périodes Δ ($\tau = \gamma\Delta$ avec $\gamma \in \mathbb{Z}$) est équivalent à un décalage de la numérotation des périodes (d'un facteur γ).
  Afin de pouvoir décrire ce phénomène, plusieurs concepts liés à la faisabilité d'une solution en lien avec un décalage temporel seront proposés ultérieurement (cf. section 3.1).
- De par la nature des contraintes de ressources agrégées par période, le PARCPSP est une relaxation du RCPSP.
  Cela reste vrai, même lorsque la durée des périodes est unitaire. L'écart entre les valeurs optimales peut même être important ; dans cet exemple, relâcher les contraintes de ressources permet de réduire de moitié la durée optimale du projet. Nous montrerons par la suite qu'il est en fait impossible de borner l'écart entre les valeurs optimales des deux problèmes, même avec des périodes de durée arbitrairement petite (cf. propositions 3.9 et 3.10).

Exemple 2 Considérons le projet représenté dans la figure 2.9 page 31, qui comporte trois activités et une ressource. Nous allons nous intéresser ici à la faisabilité de la solution $S = (\tau, \tau+1, \tau+4)$, avec $\tau = 0$ (cf. figure 2.10 page 32), $\tau = 1$ (cf. figure 2.11 page 33), ou $\tau = 2$ (cf. figure 2.12 page 34), pour le PARCPSP défini avec une subdivision temporelle uniforme de période $\Delta \in \{2,3,4,5\}$; Le tableau 2.1 indique pour chaque cas si la solution S est réalisable ou non.

Ce second exemple permet donc de montrer que la faisabilité est impactée de manière non régulière non seulement par un décalage temporel τ, mais aussi par la durée des périodes Δ. On peut notamment remarquer que, pour $\tau = 0$, S est réalisable si $\Delta = 3$ mais irréalisable si $\Delta = 5$; c'est le contraire pour $\tau = 1$.

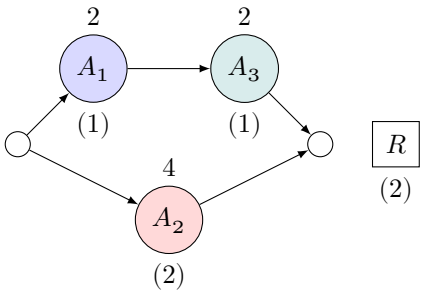


Fig. 2.9 : Exemple #2 – Projet

	$\tau = 0$	$\tau = 1$	$\tau = 2$
$\Delta = 2$	Réalisable	Irréalisable	Réalisable
$\Delta = 3$	Réalisable	Irréalisable	Irréalisable
$\Delta = 4$	Réalisable	Réalisable	Réalisable
$\Delta = 5$	Irréalisable	Réalisable	Réalisable

Tab. 2.1 : Exemple #2 – Synthèse

Par ailleurs, augmenter progressivement la valeur de Δ ne signifie pas relâcher de plus en plus les contraintes de ressources agrégées (cf. colonne $\tau = 0$ ou $\tau = 2$), à moins de considérer des valeurs multiples, comme c'est le cas ici avec $\Delta = 2$ et $\Delta = 4$. Ces phénomènes seront étudiées plus en détail dans la section 3.3.

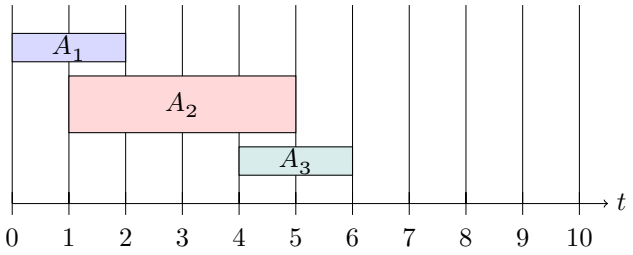
De plus, pour $\Delta = 2$, on peut remarquer que $S = (0, 1, 4)$ (i.e. $\tau = 0$) est une solution réalisable, et même optimale, qui intersecte seulement trois périodes, $[0; 2]$, $[2; 4]$ et $[4; 6]$, tandis que $S = (1, 2, 5)$ (i.e. $\tau = 1$) est une solution non réalisable qui intersecte une période supplémentaire, $[6; 8]$. Cela montre que répartir l'exécution du projet sur un plus grand nombre de périodes n'est pas toujours une stratégie gagnante pour construire une solution optimale.

2.3 Complexité

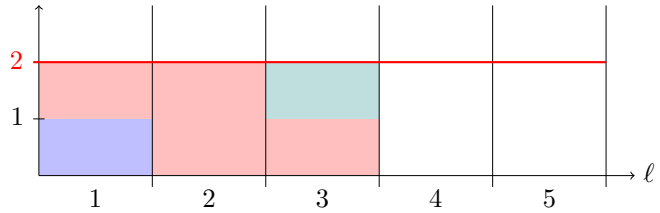
Cette section est dédiée à l'analyse de la difficulté du PARCPSP dans le cadre de la théorie de la complexité. Tout d'abord, l'appartenance du problème de décision associé au PARCPSP à la classe des problèmes non déterministes polynomiaux (**NP**) est démontrée (cf. proposition 2.2). Ensuite, la **NP**-complétude de ce problème est établie dans trois cas distincts (restriction sur les données).

L'étude de complexité a été présentée dans une conférence (Morin et al. 2018b) et un rapport technique (Morin, Artigues, Hait et al. 2017) en collaboration avec les professeurs Frits Spieksma et Tamás Kis.

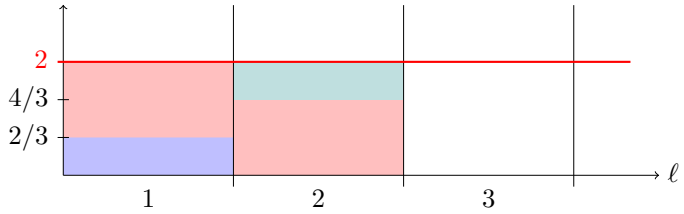
1. Nombre de ressources fixé et capacités limitées : **NP**-complet au sens faible (réduction au problème Partition, cf. proposition 2.3, objectif S_{n+1}).
   Ce résultat a pu être obtenu grâce à une collaboration avec le professeur Tamás Kis.
2. Nombre de ressources fixé et capacités illimitées : **NP**-complet au sens fort



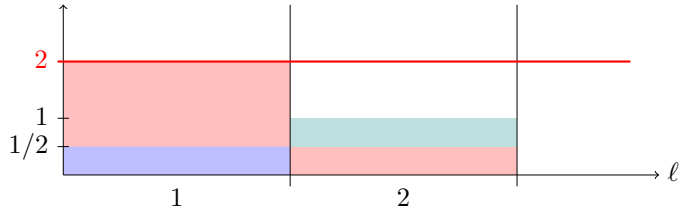
(a) Solution



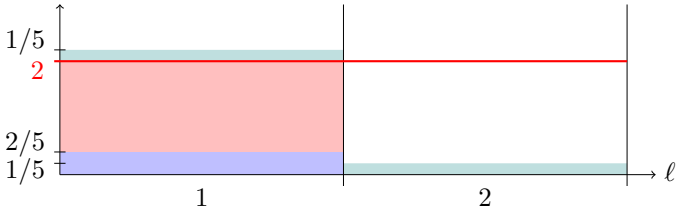
(b) Profil de consommation agrégé sur des périodes de durée $\Delta = 2$



(c) Profil de consommation agrégé sur des périodes de durée $\Delta = 3$

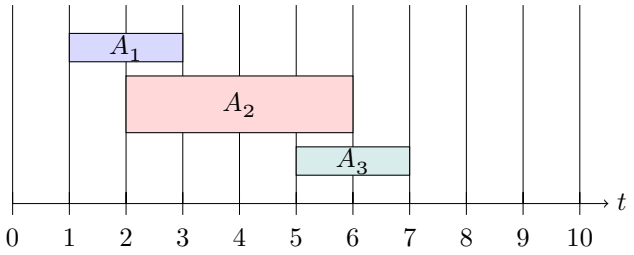


(d) Profil de consommation agrégé sur des périodes de durée $\Delta = 4$

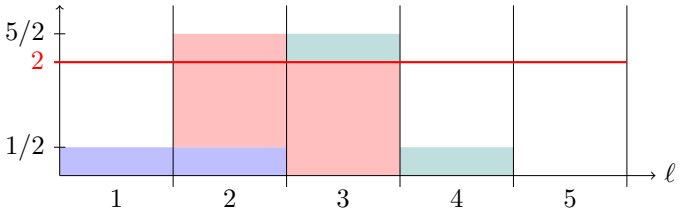


(e) Profil de consommation agrégé sur des périodes de durée $\Delta = 5$

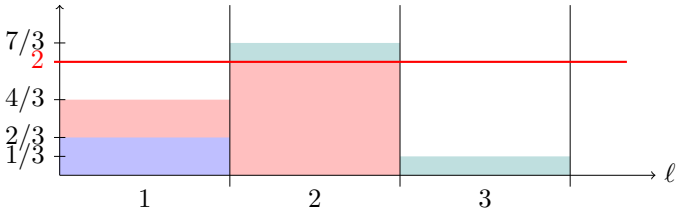
Fig. 2.10 : Exemple #2 – Solution #1



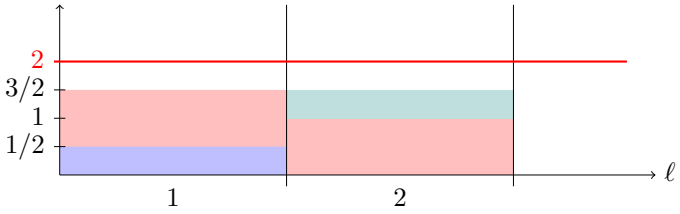
(a) Solution



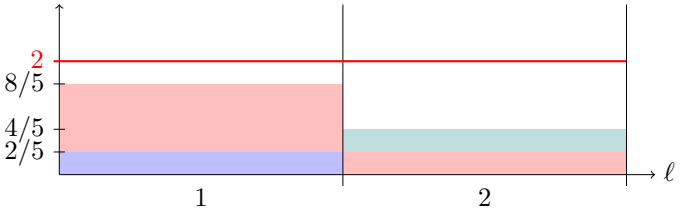
(b) Profil de consommation agrégé sur des périodes de durée $\Delta = 2$



(c) Profil de consommation agrégé sur des périodes de durée $\Delta = 3$

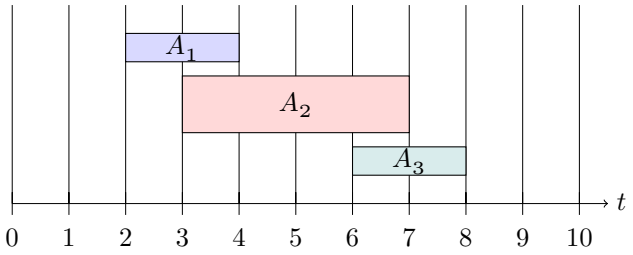


(d) Profil de consommation agrégé sur des périodes de durée $\Delta = 4$

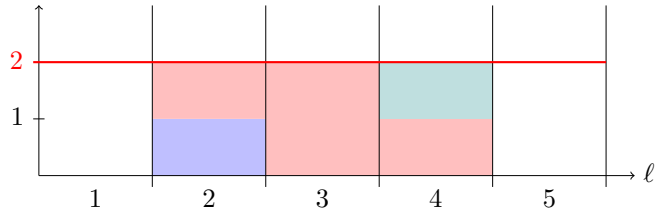


(e) Profil de consommation agrégé sur des périodes de durée $\Delta = 5$

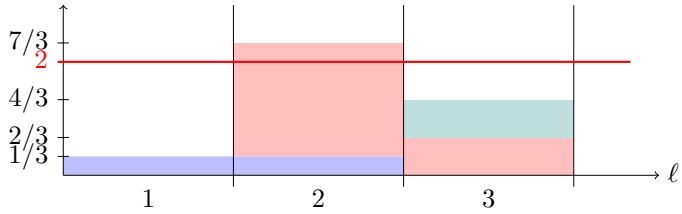
Fig. 2.11 : Exemple #2 – Solution #2



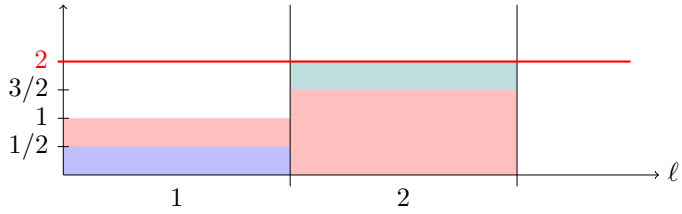
(a) Solution



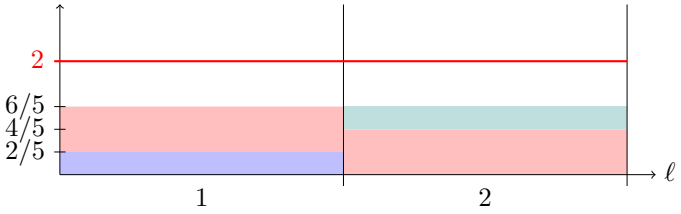
(b) Profil de consommation agrégé sur des périodes de durée $\Delta = 2$



(c) Profil de consommation agrégé sur des périodes de durée $\Delta = 3$



(d) Profil de consommation agrégé sur des périodes de durée $\Delta = 4$



(e) Profil de consommation agrégé sur des périodes de durée $\Delta = 5$

Fig. 2.12 : Exemple #2 – Solution #3

(réduction au problème 3-Partition, cf. proposition 2.4, objectif S_{n+1}).
Ce résultat a pu être obtenu grâce à une collaboration avec le professeur Frits Spieksma.

3. Nombre de ressources non fixé et capacités limitées : **NP**-complet au sens fort (réduction au problème de coloration de graphe, cf. proposition 2.5, objectif quelconque).

Le PARCPSP est également **NP**-difficile au sens fort dans le cas où le nombre de ressources n'est pas fixé et les capacités sont illimitées (généralisation des cas 2 et 3). Ces résultats montrent que le PARCPSP est un problème combinatoire difficile.

2.3.1 Appartenance à la classe NP

Proposition 2.2. *PARCPSP* $\in$ **NP**

Démonstration. L'algorithme 2.1 page 36 vérifie la faisabilité d'un ordonnancement en temps polynomial.

1. Le test de la durée de projet peut être effectué en un temps linéaire en le nombre d'activités.
2. Le test de satisfaction des contraintes d'antériorité consiste à vérifier que, pour chaque couple prédécesseur/successeur (arc) du graphe d'antériorité, la date de fin du premier est inférieure à la date de début du second.
3. Concernant la satisfaction des contraintes de ressources agrégées, il n'est pas possible de réaliser un test par ressource et par période, car cela aboutirait à un algorithme non-polynomial (le nombre de périodes n'est pas une donnée du problème). Cependant, pour chaque ressource et chaque activité, il suffit de tester les contraintes de ressources agrégées dans (au plus) deux périodes : celle dans laquelle l'activité commence (toujours), et la période suivante (parfois). En effet, étant donnée une activité $i \in \mathcal{A}$, puisque la préemption n'est pas autorisée, la valeur de $D_{i,\ell}(S_i)$ ne peut augmenter que dans ces deux périodes, i.e., la différence $D_{i,\ell}(S_i) - D_{i,\ell-1}(S_i)$ ne peut être strictement positive que pour $\ell \in \left\{\ell_i^S, \ell_i^S + 1\right\}$ où ℓ_i^S désigne une période qui contient S_i. De plus, un test sur une ressource donnée dans une période donnée peut être effectué en un temps linéaire en fonction du nombre d'activités (calcul de la somme des consommations moyennes).

Finalement, la complexité temporelle de cet algorithme est donc $\mathcal{O}(mn^2)$. □

2.3.2 Réduction depuis le problème Partition

Nous allons maintenant montrer que le PARCPSP est NP-difficile (au moins au sens faible) lorsque le nombre de ressources est fixé (dont la valeur est bornée par une constante numérique) et les capacités des ressources sont limitées (i.e. bornées par une constante numérique). La fonction objectif considérée ici est S_{n+1}.

Proposition 2.3. *Partition* $\propto$ *PARCPSP*

Démonstration. Le problème Partition peut être défini de la manière suivante.

> *Soit I un ensemble de n éléments ($n \in \mathbb{N}$). À chaque élément $i \in I$ est associé un poids $w_i \in \mathbb{N}^*$. On suppose que la somme des poids $W = \sum_{i \in I} w_i$ est un entier pair. Est-il possible de partitionner I en deux sous-ensembles A et B de même poids $\frac{W}{2}$?*

Données : Un projet $X \in \mathcal{X}$; une subdivision temporelle uniforme caractérisée par des périodes de même durée $\Delta \in \mathbb{R}_+^*$; une alternative $S \in \mathbb{R}^{\mathcal{A}}$; une borne supérieure du critère $h \in \mathbb{R}_+^*$.
Résultat : `Vrai` si S est une solution telle que $S_{n+1} - S_0 \leq h$; `Faux` sinon.

$S_0 \leftarrow \min_{i \in \mathcal{A}} (S_i)$
$S_{n+1} \leftarrow \max_{i \in \mathcal{A}} (S_i + p_i)$
si $\neg (S_{n+1} - S_0 \leq h)$ **alors**
  retourner `Faux`
pour chaque $(i_1, i_2) \in E$ **faire** /* $|E| \leq |\mathcal{A}|^2$ *itérations* */
  si $\neg \left(S_{i_1} + p_{i_1} \leq S_{i_2} \right)$ **alors**
    retourner `Faux`
pour chaque $k \in \mathcal{R}$ **faire** /* $|\mathcal{R}|$ *itérations* */
  pour chaque $i \in \mathcal{A}$ **faire** /* $|\mathcal{A}|$ *itérations* */
    $\ell \leftarrow \left\lfloor \frac{S_i}{\Delta} \right\rfloor$
    $a \leftarrow 0$
    pour chaque $i' \in \mathcal{A}$ **faire** /* $|\mathcal{A}|$ *itérations* */
      $a \leftarrow a + r_{i',k} \, D_{i',\ell}(S_{i'})$
    /* $a = \sum_{i' \in \mathcal{A}} r_{i',k} \, D_{i',\ell}(S_{i'})$ */
    si $\neg (a \leq b_k \, \Delta)$ **alors**
      retourner `Faux`
    $a \leftarrow 0$
    pour chaque $i' \in \mathcal{A}$ **faire** /* $|\mathcal{A}|$ *itérations* */
      $a \leftarrow a + r_{i',k} \, D_{i',\ell+1}(S_{i'})$
    /* $a = \sum_{i' \in \mathcal{A}} r_{i',k} \, D_{i',\ell+1}(S_{i'})$ */
    si $\neg (a \leq b_k \, \Delta)$ **alors**
      retourner `Faux`
retourner `Vrai`

Algorithme 2.1 : Vérificateur de solution pour le PARCPSP (appartenance à **NP**)

Étant donnée une instance du problème Partition, soit l'instance du problème PARCPSP définie de la manière suivante :

- $\mathcal{A} = I$
- $|\mathcal{R}| = 1$
- $\Delta = 1$
- $\forall i \in \mathcal{A} \qquad p_i = 2\,w_i\,\Delta = 2\,w_i$
- $b_1 = 2$
- $\forall i \in \mathcal{A} \qquad r_{i,1} = 1$
- $E = \emptyset$ (aucune relation d'antériorité)

Il est clair que cette transformation est une réduction polynomiale. Par conséquent, Partition $\propto$ PARCPSP si les deux propositions suivantes sont équivalentes.

[1] L'instance du problème Partition admet une solution (partition en deux sous-ensembles A et B de même poids $\frac{W}{2}$).

[2] L'instance du problème PARCPSP admet une solution (ordonnancement réalisable) S telle que : $S_0 = 0$ et $S_{n+1} = \frac{1}{2}\sum_{i\in\mathcal{A}} p_i$

[1] $\Rightarrow$ [2]

Par hypothèse [1], il existe deux sous-ensembles A et B de I tels que :

- $A \cap B = \emptyset$
- $A \cup B = I$
- $\sum_{i\in A} w_i = \sum_{i\in B} w_i = \frac{W}{2}$

Soit S l'ordonnancement construit de la manière suivante.

- Les activités associées aux éléments dans A sont exécutées dans un ordre quelconque, à partir de la date $t = 0$, sans temps mort, donc jusqu'à la date $t = \sum_{i\in A} p_i$. Remarquons que chaque activité de cette sous-séquence commence et termine à un instant entier, i.e. une date qui coïncide avec une borne temporelle (multiple de $\Delta = 1$) ; de plus, à chaque instant, la consommation est égale à 1.
- De même, les activités associées aux éléments dans B sont exécutées dans un ordre quelconque, à partir de la même date $t = 0$, sans temps mort, donc jusqu'à la même date $t = \sum_{i\in B} p_i$; chaque activité de cette sous-séquence commence et termine à un instant entier et, à chaque instant, la consommation est égale à 1.

Par conséquent, l'ordonnancement commence à la date $S_0 = 0$ et termine à la date $S_{n+1} = \frac{1}{2}\sum_{i\in I} p_i$. À chaque instant, la consommation est égale à 2 ; dans chaque période de durée unitaire ($\Delta = 1$), la consommation moyenne est donc égale à 2, ce qui n'excède pas la capacité de l'unique ressource ($= 2$). Par conséquent, [2] est vérifiée.

[2] $\Rightarrow$ [1]

Par hypothèse [2], il existe un ordonnancement réalisable S tel que $S_0 = 0$ et $S_{n+1} = \frac{1}{2}\sum_{i\in\mathcal{A}} p_i = W$.

Lemme 2.3a. *Dans toute période unitaire $[t-1, t]$, pour $t \in \{1, \ldots, W\}$, la consommation moyenne est exactement égale à 2.*

Démonstration. Puisque S est réalisable, la consommation moyenne dans toute période (de durée unitaire $\Delta = 1$) est inférieure ou égale à 2 (capacité de la ressource).

Par ailleurs, la quantité totale de ressource disponible entre S_0 et S_{n+1} est égale à $2(S_{n+1} - S_0) = 2W$, tandis que la quantité totale de ressource consommée par l'ensemble des activités (dont la consommation est unitaire) est égale à $\sum_{i \in \mathcal{A}} p_i = 2W$. Par conséquent, la consommation moyenne dans toute période unitaire entre S_0 et S_{n+1} est supérieure ou égale à 2. ■

Lemme 2.3b. *Exactement deux activités commencent à la date $t = 0$.*

Démonstration. S'il existait au moins trois activités commençant dans la période $[0, 1]$, puisque les durées d'exécution sont des entiers pairs ($p_i = 2w_i$) et les périodes sont de durée unitaire ($\Delta = 1$), les fenêtres d'exécution de toutes ces activités termineraient dans la période $[2, 3]$, donc contiendraient la période $[1, 2]$ dans sa totalité. Or, cela impliquerait que la consommation moyenne dans la période $[1, 2]$ soit supérieure ou égale à 3, ce qui est impossible d'après le lemme 2.3a. Par conséquent, au plus deux activités commencent dans la période $[0, 1]$.

De plus, si le nombre d'activités commençant à la date 0 était strictement inférieur à 2, alors la consommation moyenne dans la période $[0, 1]$ serait strictement inférieure à 2, ce qui est impossible, toujours d'après le lemme 2.3a. ■

Au début de l'ordonnancement S, il existe donc exactement deux activités qui commencent à la date 0. Puisque les durées d'exécution sont entières, et les périodes sont de durée unitaire, ces activités terminent à des instants entiers, t_1 et t_2 (quitte à les intervertir, supposons que $t_1 \leq t_2$).

- $t_1 = t_2$
  En appliquant le même raisonnement que celui utilisé dans la preuve du lemme 2.3b, on montre que deux nouvelles activités commencent simultanément à cet instant.
- $t_1 < t_2$
  Puisque les durées d'exécution sont paires : $t_2 \geq t_1 + 2$. Avec un raisonnement similaire, on montre d'abord qu'au plus une activité peut commencer dans la période $[t_1, t_1 + 1]$, puis qu'une et une seule activité doit commencer son exécution à la date t_1.

En continuant ainsi, on montre que la structure de S vérifie les propriétés suivantes :

- Dès qu'une activité termine, une autre commence (sauf aux bords, i.e. $t = 0$ et $t = S_{n+1}$).
- À chaque instant entre S_0 et S_{n+1}, exactement deux activités sont en cours d'exécution.

Par conséquent, l'algorithme 2.2 page 39 permet d'extraire, en temps polynomial, une partition de l'ensemble des activités/items de même durée/poids (version détaillée avec assertions ; l'algorithme 2.3 page 40 en est une version simplifiée mais néanmoins équivalente). □

2.3.3 Réduction depuis le problème 3-Partition

En s'inspirant de la réduction précédente, nous allons maintenant montrer que le PARCPSP est NP-difficile (au sens fort) lorsque le nombre de ressources est fixé (dont la valeur est bornée par une constante numérique) et les capacités des ressources sont illimitées (i.e. non bornées par une constante numérique). La fonction objectif considérée ici est également S_{n+1}.

Données :

- Une instance du PARCPSP obtenue à partir d'une instance du problème Partition (cf. réduction polynomiale définie page 37) ;
- Une solution $S \in \mathbb{R}^{\mathcal{A}}$ (réalisable) telle que :
  - $S_0 = 0$
  - $S_{n+1} = \frac{1}{2}\sum_{i\in\mathcal{A}} p_i = W$

  Les activités $i \in \mathcal{A}$ doivent être triées par dates de début (S_i) croissantes.

Résultat :

- Deux sous-ensembles d'activités A et B tels que :
  - $A \cap B = \emptyset$
  - $A \cup B = \mathcal{A}$
  - $\sum_{i\in A} p_i = \sum_{i\in B} p_i = \frac{1}{2}\sum_{i\in\mathcal{A}} p_i = W$

```
1  assert : S_0 = S_1 = S_2 = 0
2  A ← {1}
3  Σ_A ← p_1                          // Somme des durées des activités insérées dans A
4  B ← {2}
5  Σ_B ← p_2                          // Somme des durées des activités insérées dans B
6  i ← 3
7  tant que i ≤ n faire
8      si Σ_A < Σ_B alors
9          assert : S_i = Σ_A
10         A ← A ∪ {i}
11         Σ_A ← Σ_A + p_i
12         i ← i + 1
13     sinon si Σ_A > Σ_B alors
14         assert : S_i = Σ_B
15         B ← B ∪ {i}
16         Σ_B ← Σ_B + p_i
17         i ← i + 1
18     sinon // Σ_A = Σ_B
19         assert : i + 1 ≤ n
20         assert : S_i = S_{i+1} = Σ_A = Σ_B
           /* L'activité i est insérée dans A, et l'activité i + 1 est insérée dans B ;
              il aurait été possible d'insérer i dans B, et i + 1 dans A. */
21         A ← A ∪ {i}
22         Σ_A ← Σ_A + p_i
23         B ← B ∪ {i + 1}
24         Σ_B ← Σ_B + p_{i+1}
25         i ← i + 2
26 assert : i = n + 1
27 assert : Σ_1 = Σ_2 = S_{n+1}
```

Algorithme 2.2 : Extraction d'une partition en deux sous-ensembles d'activités/items de même durée/poids (version détaillée)

Données :

- Une instance du PARCPSP obtenue à partir d'une instance du problème Partition (cf. réduction polynomiale définie page 37)) ;
- Une solution $S \in \mathbb{R}^{\mathcal{A}}$ (réalisable) telle que :
  - $S_0 = 0$
  - $S_{n+1} = \frac{1}{2} \sum_{i \in \mathcal{A}} p_i = W$

  Les activités $i \in \mathcal{A}$ doivent être triées par dates de début (S_i) croissantes.

Résultat :

- Deux sous-ensembles d'activités A et B tels que :
  - $A \cap B = \emptyset$
  - $A \cup B = \mathcal{A}$
  - $\sum_{i \in A} p_i = \sum_{i \in B} p_i = \frac{1}{2} \sum_{i \in \mathcal{A}} p_i = W$

```
1  A ← ∅
2  Σ_A ← 0                          // Somme des durées des activités insérées dans A
3  B ← ∅
4  Σ_B ← 0                          // Somme des durées des activités insérées dans B
5  pour i = 1 à n faire
6      si Σ_A ≤ Σ_B alors
7          A ← A ∪ {i}
8          Σ_A ← Σ_A + p_i
9      sinon
10         B ← B ∪ {i}
11         Σ_B ← Σ_B + p_i
```

Algorithme 2.3 : Extraction d'une partition en deux sous-ensembles d'activités/items de même durée/poids (version simplifiée)

Proposition 2.4. *3-Partition* $\propto$ *PARCPSP*

Démonstration. Le problème 3-Partition peut être défini de la manière suivante.

> *Soit I un ensemble de $3n$ éléments ($n \in \mathbb{N}$). À chaque élément $i \in I$ est associé un poids $w_i \in \mathbb{N}^*$. On suppose que la somme des poids $W = \sum_{i \in I} w_i$ est un multiple de n et que, pour chaque élément $i \in I$, $\frac{W}{4n} < w_i < \frac{W}{2n}$. Est-il possible de partitionner I en n triplets $T_1, T_2, \dots, T_n$ de même poids $\frac{W}{n}$?*

Étant donnée une instance du problème 3-Partition, soit l'instance du problème PARCPSP définie de la manière suivante :

- $\mathcal{A} = I$
- $|\mathcal{R}| = 1$
- $\Delta = 1$
- $\forall i \in \mathcal{A} \qquad p_i = 2\, w_i\, \Delta = 2\, w_i$
- $b_1 = n$
- $\forall i \in \mathcal{A} \qquad r_{i,1} = 1$
- $E = \emptyset$ (aucune relation d'antériorité)

Il est clair que cette transformation est une réduction polynomiale. Par conséquent, 3-Partition $\propto$ PARCPSP si les deux propositions suivantes sont équivalentes.

[1] L'instance du problème 3-Partition admet une solution (partition en n triplets $T_1, T_2, \dots, T_n$ de même poids $\frac{W}{n}$).

[2] L'instance du problème PARCPSP admet une solution (ordonnancement réalisable) S telle que : $S_0 = 0$ et $S_{n+1} = \frac{1}{n} \sum_{i \in \mathcal{A}} p_i$

[1] $\Rightarrow$ [2]

Par hypothèse [1], il existe n triplets $T_1, T_2, \dots, T_n$ de même poids $\frac{W}{n}$ qui forment une partition de I.

Soit S l'ordonnancement construit de la manière suivante. Pour chaque triplet T, les activités associées aux éléments dans ce triplet sont exécutées les unes à la suite des autres dans un ordre quelconque, à partir de la date $t = 0$, sans temps mort, donc jusqu'à la date $t = \sum_{i \in T} p_i$. Remarquons que chaque activité de cette sous-séquence commence et termine à un instant entier, i.e. une date qui coïncide avec une borne temporelle (multiple de $\Delta = 1$) ; de plus, à chaque instant, la consommation est égale à 1.

Par conséquent, l'ordonnancement commence à la date $S_0 = 0$ (date de début commune à tous les triplets) et termine à la date $S_{n+1} = \frac{1}{n} \sum_{i \in I} p_i$ (date de fin commune à tous les triplets). À chaque instant, la consommation est égale à n ; dans chaque période de durée unitaire ($\Delta = 1$), la consommation moyenne est donc égale à n, ce qui n'excède pas la capacité de l'unique ressource ($= n$). Par conséquent, [2] est vérifié.

[2] $\Rightarrow$ [1]

Par hypothèse [2], il existe un ordonnancement réalisable S tel que $S_0 = 0$ et $S_{n+1} = \frac{1}{n} \sum_{i \in \mathcal{A}} p_i = \frac{2W}{n}$.

Les lemmes 2.3a et 2.3b peuvent être transposés de la manière suivante.

Lemme 2.4a. *Dans toute période unitaire* $[t-1, t]$*, pour* $t \in \left\{1, \dots, \frac{2W}{n}\right\}$*, la consommation moyenne est exactement égale à* n*.*

Lemme 2.4b. *Exactement* n *activités commencent à la date* $t = 0$*.*

De plus, il ne peut exister d'intervalle temporel $[t-1, t]$ ($t \in \left\{1, \dots, \frac{2W}{n}\right\}$) qui contiendrait un instant auquel (au plus) $n-1$ activités sont en cours d'exécution. Si tel était le cas, dans une telle période, puisque la consommation moyenne est égale à n, il existerait un autre instant auquel (au moins) $n+1$ activités seraient en cours d'exécution. Or, les durées d'exécution étant des entiers pairs et les périodes de durée unitaire, cela impliquerait qu'une période voisine serait incluse en totalité dans la fenêtre d'exécution de ces $n+1$ activités, ce qui est impossible d'après le lemme 2.4a.

Par conséquent, à chaque instant entre $S_0 = 0$ et $S_{n+1} = \frac{1}{n}\sum_{i \in \mathcal{A}} p_i$, exactement n activités sont en cours d'exécution. S est donc structuré de telle manière qu'il est possible d'extraire n sous-séquences/sous-ensembles de même durée/poids $\frac{W}{n}$ formant une partition de l'ensemble des activités/éléments. Puisque, pour tout élément $i \in I$, $\frac{W}{4n} < w_i < \frac{W}{2n}$, chacun de ces sous-ensembles est nécessairement un triplet. □

2.3.4 Réduction depuis le problème de coloration des sommets d'un graphe

En utilisant une réduction polynomiale à partir d'un autre problème, nous allons maintenant montrer que le PARCPSP est NP-difficile (au sens fort) lorsque le nombre de ressources est arbitraire (paramètre fixe, mais dont la valeur n'est pas bornée par une constante numérique) et les capacités des ressources sont limitées (i.e. bornées par une constante numérique). Cette preuve, présentée ici avec la fonction objectif $S_{n+1} - S_0$, reste néanmoins valide pour d'autres fonctions objectifs (comme S_{n+1}, considérée précédemment), car le lemme 2.5a ne dépend pas du choix de la fonction objectif.

Proposition 2.5. *Coloration des sommets d'un graphe* $\propto$ *PARCPSP*

Démonstration. Le problème de coloration des sommets d'un graphe peut être défini de la manière suivante.

> *Soit* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *un graphe non-orienté contenant* $n = |\mathcal{V}| \in \mathbb{N}$ *sommets et* $m = |\mathcal{E}| \in \mathbb{N}$ *arêtes. Soit* $\gamma \in \mathbb{N}$*. Existe-t-il une coloration des sommets du graphe* $\mathcal{G}$ *utilisant au plus* γ *couleurs telle que, pour toute arête, ses deux extrémités ont une couleur différente ?*

Étant donnée une instance du problème de coloration des sommets d'un graphe, soit l'instance du problème PARCPSP définie de la manière suivante :

- $\mathcal{A} = \mathcal{V}$
- $\mathcal{R} = \mathcal{E}$
- $\Delta = 1$
- $\forall i \in \mathcal{A} \qquad p_i = 2\,\Delta = 2$
- $\forall k \in \mathcal{R} \qquad b_k = 1$
- $\forall i \in \mathcal{A} \quad \forall k \in \mathcal{R} \qquad r_{i,k} = 1$ si le sommet i est l'une des deux extrémités de l'arête k, 0 sinon

- $E = \emptyset$ (aucune relation d'antériorité)

Il est clair que cette transformation est une réduction polynomiale. Par conséquent, Coloration de graphe $\propto$ PARCPSP si les deux propositions suivantes sont équivalentes.

[1] L'instance du problème de coloration de graphe admet une coloration réalisable $c \in (\mathbb{N}^*)^n$ telle que : $\max_{i \in \mathcal{V}} c_i \leq \gamma$

[2] L'instance du problème PARCPSP admet une solution (ordonnancement réalisable) S telle que : $S_{n-1} - S_0 \leq 2\gamma$

[1] $\Rightarrow$ [2]

Par hypothèse [1], il existe une coloration réalisable $c \in (\mathbb{N}^*)^n$ telle que :

$$\forall i \in \mathcal{V} \quad 1 \leq c_i \leq \gamma$$

Soit S l'ordonnancement défini par : $\forall i \in \mathcal{A} \quad S_i = 2(c_i - 1)\Delta$

Étant donnée une arête (ressource), ses extrémités (les deux seules activités qui ont une consommation non-nulle sur cette ressource) sont coloriées différemment (ont des fenêtres d'exécution disjointes). S est donc réalisable pour le PARCPSP (et même pour le RCPSP).

De plus : $S_{n+1} - S_0 \leq 2\gamma\Delta - 0 = 2\gamma\Delta$

Par conséquent, [2] est vérifié.

[2] $\Rightarrow$ [1]

Par hypothèse [2], il existe un ordonnancement réalisable S tel que $S_{n+1} - S_0 \leq 2\gamma\Delta$. Sans perte de généralité, on suppose que l'exécution du projet commence dans la première période : $0 \leq S_0 < \Delta$

Lemme 2.5a. *Les fenêtres d'exécution de deux activités ayant une consommation non nulle sur une même ressource sont disjointes.*

$$\forall (i_1, i_2) \in \mathcal{R} \quad \left(S_{i_1} + p_{i_1} \leq S_{i_2}\right) \vee \left(S_{i_2} + p_{i_2} \leq S_{i_1}\right)$$

Démonstration. Soit $k = (i_1, i_2) \in \mathcal{R}$. Sans perte de généralité, on suppose que $S_{i_1} \leq S_{i_2}$.

Pour toute activité $i \in \mathcal{A}$, soit $\ell_i = 1 + \left\lfloor \frac{S_i}{\Delta} \right\rfloor$ l'indice de la période dans laquelle commence l'activité i. Remarquons que $\ell_{i_1} \leq \ell_{i_2}$.

Pour toute activité $i \in \mathcal{A}$, dans toute période $\ell \in \mathbb{Z}$, puisque $p_i = 2\Delta$, il est possible de déterminer des bornes sur les valeurs $D_{i,\ell}(S_i)$ (cf. figure 2.13) :

$$D_{i,\ell}(S_i) \begin{cases} \in]0, \Delta] & \text{si } \ell = \ell_i \\ = \Delta & \text{si } \ell = \ell_i + 1 \\ \in [0, \Delta[& \text{si } \ell = \ell_i + 2 \\ = 0 & \text{sinon} \end{cases}$$

Plus précisément : $D_{i,\ell_i+2}(S_i) = \Delta - D_{i,\ell_i}(S_i)$

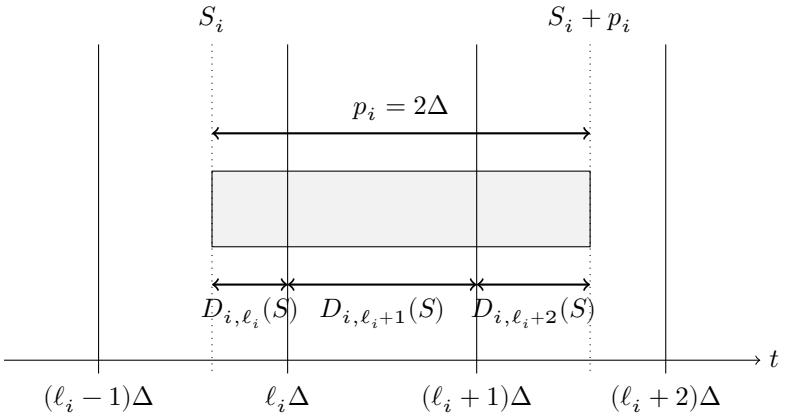


Fig. 2.13 : Positionnement d'une fenêtre d'exécution de longueur fixe ($p_i = 2\Delta$) par rapport aux périodes

Rappelons que S est un ordonnancement réalisable, donc vérifie les contraintes de ressources agrégées par période. Puisque la capacité de la ressource est unitaire, et les consommations des activités sont unitaires ou nulles, la contrainte associée à la ressource/arête (i_1, i_2) s'exprime sous la forme suivante :

$$D_{i_1,\ell}(S_{i_1}) + D_{i_2,\ell}(S_{i_2}) \leq \Delta \qquad \forall \ell \in \mathbb{Z}$$

- Supposons que : $\ell_{i_2} = \ell_{i_1}$
  Dans ce cas, dans la période $\ell = \ell_{i_1} + 1 = \ell_{i_2} + 1$:
  $D_{i_1,\ell}(S_{i_1}) + D_{i_2,\ell}(S_{i_2}) = 2\Delta > \Delta$
  Cette configuration est donc impossible (puisque S est réalisable).
- Supposons que : $\ell_{i_2} = \ell_{i_1} + 1$
  Dans ce cas, dans la période $\ell = \ell_{i_1} + 1 = \ell_{i_2}$:
  $D_{i_1,\ell}(S_{i_1}) + D_{i_2,\ell}(S_{i_2}) = \Delta + D_{i_2,\ell}(S_{i_2}) > \Delta$
  Cette configuration est donc impossible.
- Supposons que : $\ell_{i_2} = \ell_{i_1} + 2$
  Dans ce cas, dans la période $\ell = \ell_{i_1} + 2 = \ell_{i_2}$:

$$\begin{aligned}
& D_{i_1,\ell}(S_{i_1}) \leq \Delta - D_{i_2,\ell}(S_{i_2}) \\
\Leftrightarrow \quad & (\ell - 1)\Delta + D_{i_1,\ell}(S_{i_1}) \leq \ell\Delta - D_{i_2,\ell}(S_{i_2}) \\
\Leftrightarrow \quad & (\ell_{i_1} + 1)\Delta + D_{i_1,\ell_{i_1}+2}(S_{i_1}) \leq \ell_{i_2}\Delta - D_{i_2,\ell_{i_2}}(S_{i_2}) \\
\Leftrightarrow \quad & S_{i_1} + p_{i_1} \leq S_{i_2}
\end{aligned}$$

- Supposons que : $\ell_{i_2} \geq \ell_{i_1} + 3$
  Alors : $S_{i_1} + p_{i_1} < (\ell_{i_1} + 2)\Delta \leq (\ell_{i_2} - 1)\Delta \leq S_{i_2}$

Dans tous les cas : $S_{i_1} + p_{i_1} \leq S_{i_2}$ ■

Soit $c \in (\mathbb{N}^*)^n$ la coloration définie par : $\forall i \in \mathcal{V} \quad c_i = 1 + \left\lfloor \frac{S_i}{\Delta} \right\rfloor$

Rappelons que les durées d'exécution des activités sont toutes égales à 2Δ ; donc, pour toute arête $(i_1, i_2) \in \mathcal{E}$, puisque $|S_{i_2} - S_{i_1}| \geq 2\Delta$ d'après le lemme 2.5a, $c_{i_1} \neq c_{i_2}$, i.e. c est une coloration réalisable.

De plus, puisque $S_{n+1} - S_0 \leq 2\gamma\Delta$ et $0 \leq S_0 < \Delta$: $\forall i \in \mathcal{A} \quad S_0 \leq S_i \leq S_{n+1} - p_i \leq (S_0 + 2\gamma\Delta) - 2\Delta = S_0 + 2(\gamma - 1)\Delta$

Autrement dit : $\forall i \in \mathcal{V} \quad 1 \leq c_i \leq \gamma$ □

Chapitre 3

PARCPSP : propriétés structurelles

Ce chapitre établit des propriétés structurelles sur le PARCPSP. Tout d'abord, la notion de faisabilité est étendue au sens de la faisabilité locale, qui qualifie une solution non nécessairement réalisable mais pour laquelle il existe une solution réalisable par translation ; la faisabilité globale concerne une solution réalisable et qui le reste par toute translation. Ensuite, des relations entre les propriétés de faisabilité d'une solution de RCPSP et de faisabilité locale et globale des solutions du PARCPSP sont établies. Il est également montré que l'écart entre la valeur optimale du RCPSP et de celle de sa relaxation PARCPSP n'est pas borné. Par ailleurs, les relations entre les éléments d'une classe d'instances du PARCPSP définies à partir du même projet mais avec des périodes différentes sont établies, au sens où il n'existe pas de relation de relaxation entre les problèmes dont les périodes ne sont pas des multiples. Nous aboutissons ainsi à la constatation qu'en général, le PARCPSP n'est pas une bonne approximation du RCPSP, ni des PARCPSP de périodes plus petites. Enfin, nous fournissons une alternative à l'algorithme 2.1 page 36 pour déterminer, en temps polynomial, si une solution est non seulement réalisable mais aussi réalisable localement et réalisable globalement.

3.1 Extensions de la notion de faisabilité

Cette section est dédiée à la définition et à l'étude de propriétés basiques de nouveaux concepts liés à la faisabilité d'une solution d'un problème d'optimisation combinatoire. Ces concepts sont applicables à une classe de problèmes d'optimisation combinatoire, qui contient le RCPSP et le PARCPSP, mais n'est pas restreinte à ces problèmes. Ces notions seront utiles afin d'étudier des propriétés structurelles du PARCPSP dans les sections suivantes.

Plus précisément, les concepts qui vont être introduits peuvent être appliqués aux problèmes d'optimisation combinatoire dont l'espace de recherche est égal à $\mathbb{R}^d$, i.e. dont les solutions (alternatives) peuvent être représentées par des vecteurs de réels de dimension d ($d \in \mathbb{N}^*$). Soit P, P_1 et P_2 de tels problèmes.

3.1.1 Définitions

Définition 3.1 (Faisabilité)**.** Une solution $S \in \mathbb{R}^d$ de P est dite faisable ou réalisable si, et seulement si, S vérifie toutes les contraintes imposées par P.

Définition 3.2 (Opérateur de translation)**.** Soit $\oplus : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ l'opérateur de translation défini par :

$$\begin{array}{l} \forall S = (S_i)_{1 \leq i \leq d} \in \mathbb{R}^d \\ \forall \tau \in \mathbb{R} \end{array} \qquad S \oplus \tau = (S_i + \tau)_{1 \leq i \leq d}$$

Définition 3.3 (Classe de solutions)**.** Soit $S \in \mathbb{R}^d$ une solution de P. La classe de solutions $\widehat{S}$ est l'ensemble des solutions pouvant être construites à partir de S par translation, i.e. :

$$\widehat{S} = \left\{ S \oplus \tau \;\middle|\; \tau \in \mathbb{R} \right\}$$

Définition 3.4 (Faisabilité locale)**.** Une solution $S \in \mathbb{R}^d$ de P est dite faisable localement ou réalisable localement si, et seulement si, au moins un élément de la classe $\widehat{S}$ est réalisable.

Remarque. Si S est réalisable localement, on ne peut *a priori* rien dire concernant la faisabilité de S. Inversement, si S est réalisable, alors S est réalisable localement (car S est une solution réalisable qui appartient à la classe $\widehat{S}$).

Définition 3.5 (Faisabilité globale)**.** Une solution $S \in \mathbb{R}^d$ de P est dite faisable globalement ou réalisable globalement si, et seulement si, tous les éléments de la classe $\widehat{S}$ sont réalisables.

Remarque. Si S est réalisable globalement, alors S est réalisable (car tout élément de la classe $\widehat{S}$, en particulier S, est réalisable). Inversement, si S est réalisable, on ne peut *a priori* rien dire concernant la faisabilité globale de S.

Soit $\mathcal{F}(P)$ l'ensemble des solutions réalisables de P, $\mathcal{LF}(P)$ l'ensemble des solutions réalisables localement de P, et $\mathcal{GF}(P)$ l'ensemble des solutions réalisables globalement de P.

$$\forall S \in \mathbb{R}^d \qquad S \in \mathcal{LF}(P) \quad \Leftrightarrow \quad (\exists \tau \in \mathbb{R} \quad S \oplus \tau \in \mathcal{F}(P))$$

$$\forall S \in \mathbb{R}^d \qquad S \in \mathcal{GF}(P) \quad \Leftrightarrow \quad (\forall \tau \in \mathbb{R} \quad S \oplus \tau \in \mathcal{F}(P))$$

3.1.2 Propriétés

Le résultat suivant permet de formaliser les remarques précédentes.

Proposition 3.1. *La faisabilité locale est une condition nécessaire de faisabilité, tandis que la faisabilité globale est une condition suffisante de faisabilité, i.e., toute solution globalement réalisable est réalisable, tandis que toute solution réalisable est réalisable localement.*

$$\mathcal{GF}(P) \subseteq \mathcal{F}(P) \subseteq \mathcal{LF}(P)$$

Il est en fait possible d'établir des liens plus forts entre les trois concepts de faisabilité.

Proposition 3.2. *Une et une seule des deux configurations suivantes est vraie pour P.*

1. *Les trois concepts de faisabilité sont équivalents.*

$$\mathcal{GF}(P) = \mathcal{F}(P) = \mathcal{LF}(P)$$

2. *Les trois concepts de faisabilité sont différents.*

$$\mathcal{GF}(P) \subset \mathcal{F}(P) \subset \mathcal{LF}(P)$$

Démonstration. Il suffit de montrer que l'équivalence suivante est vraie.

Lemme 3.2a. $\mathcal{LF}(P) \setminus \mathcal{F}(P) \neq \emptyset \quad \Leftrightarrow \quad \mathcal{F}(P) \setminus \mathcal{GF}(P) \neq \emptyset$

Démonstration.

$$\begin{aligned}
& \mathcal{LF}(P)\backslash\mathcal{F}(P) \neq \emptyset \\
\Leftrightarrow \quad & \exists S \in \mathcal{LF}(P) \quad S \notin \mathcal{F}(P) \\
\Leftrightarrow \quad & \exists (S,\tau) \in (\mathbb{R}^d \times \mathbb{R}) \quad (S \oplus \tau \in \mathcal{F}(P)) \wedge (S \notin \mathcal{F}(P)) \\
\Leftrightarrow \quad & \exists (S',\tau') \in (\mathbb{R}^d \times \mathbb{R}) \quad (S' \in \mathcal{F}(P)) \wedge (S' \oplus \tau' \notin \mathcal{F}(P)) \\
\Leftrightarrow \quad & \exists S' \in \mathcal{F}(P) \quad S' \notin \mathcal{GF}(P) \\
\Leftrightarrow \quad & \mathcal{F}(P)\backslash\mathcal{GF}(P) \neq \emptyset
\end{aligned}$$

Remarque. De la troisième à la quatrième ligne :

$$\left\{ \begin{array}{l} S' = S \oplus \tau \\ \tau' = -\tau \end{array} \right. \Leftrightarrow \left\{ \begin{array}{l} S = S' \oplus \tau' \\ \tau = -\tau' \end{array} \right.$$

Autrement dit, S (solution irréalisable) et S' (solution réalisable) appartiennent à la même classe ($\hat{S} = \widehat{S'}$). ■

Ainsi, les ensembles $\mathcal{LF}(P) \setminus \mathcal{F}(P)$ et $\mathcal{F}(P) \setminus \mathcal{GF}(P)$ sont soit vides tous les deux, et tous les concepts sont alors équivalents (cas 1), soit non vides tous les deux, et les trois concepts sont alors deux à deux distincts (cas 2). □

Remarque. Pour le RCPSP, tous les concepts sont équivalents (cas 1), puisque la faisabilité d'un ordonnancement est invariante par translation. En revanche, pour le PARCPSP, tous les concepts sont deux à deux distincts (cas 2 ; cf. exemples de la section 2.2 pour lesquels la translation d'un ordonnancement affecte sa faisabilité).

Proposition 3.3. *Si P_2 est une relaxation de P_1, alors les relations d'inclusion entre ensembles sont valables pour les trois concepts. Si P_1 et P_2 sont des problèmes équivalents, alors les relations d'égalité entre ensembles sont valables pour les trois concepts.*

$$\mathcal{F}(P_1) \subseteq \mathcal{F}(P_2) \quad \Rightarrow \quad \left\{ \begin{array}{l} \mathcal{LF}(P_1) \subseteq \mathcal{LF}(P_2) \\ \mathcal{GF}(P_1) \subseteq \mathcal{GF}(P_2) \end{array} \right.$$

$$\mathcal{F}(P_1) = \mathcal{F}(P_2) \quad \Rightarrow \quad \left\{ \begin{array}{l} \mathcal{LF}(P_1) = \mathcal{LF}(P_2) \\ \mathcal{GF}(P_1) = \mathcal{GF}(P_2) \end{array} \right.$$

Démonstration. Supposons que $\mathcal{F}(P_1) \subseteq \mathcal{F}(P_2)$.

- Soit $S \in \mathcal{LF}(P_1)$: il existe $\tau \in \mathbb{R}$ tel que $S \oplus \tau \in \mathcal{F}(P_1)$. Par conséquent, $S \oplus \tau \in \mathcal{F}(P_2)$; d'où $S \in \mathcal{LF}(P_2)$.
- Soit $S \in \mathcal{GF}(P_1)$: pour tout $\tau \in \mathbb{R}$, $S \oplus \tau \in \mathcal{F}(P_1)$. Par conséquent, pour tout $\tau \in \mathbb{R}$, $S \oplus \tau \in \mathcal{F}(P_2)$; d'où $S \in \mathcal{GF}(P_2)$.

La première affirmation est donc vraie ; la seconde en découle par symétrie. □

3.2 Propriétés structurelles du PARCPSP en tant que relaxation du RCPSP

Il est aisé de remarquer que le PARCPSP est une relaxation du RCPSP. De ce fait, il est légitime d'étudier qualitativement cette relaxation, afin d'exhiber les ressemblances et les différences entre les deux problèmes.

Dans cette section sont présentées les propriétés suivantes :

- démonstration formelle du fait que le PARCPSP est une relaxation du RCPSP (proposition 3.4)
- établissement d'un lien entre les solutions réalisables du RCPSP et les solutions globalement réalisables du PARCPSP (proposition 3.5)
- existence, pour toute solution non réalisable du RCPSP, d'une valeur seuil pour la durée des périodes en dessous de laquelle cette même solution n'est pas réalisable (même localement) pour le PARCPSP (proposition 3.6)
- gap absolu non borné, même avec une seule ressource de capacité constante (proposition 3.9 qui découle des propositions 3.7 et 3.8)
- gap relatif non borné, même avec plusieurs ressources de capacités constantes (proposition 3.10)

Proposition 3.4. *Le PARCPSP est une relaxation du RCPSP.*

$$\begin{cases} \forall X \in \mathcal{X} \\ \forall \Delta \in \mathbb{R}_+^* \end{cases} \qquad \mathcal{F}(RCPSP[X]) \subseteq \mathcal{F}(PARCPSP[X, \Delta])$$

Démonstration. Soit $X \in \mathcal{X}$. Soit $\Delta \in \mathbb{R}_+^*$. Soit $S \in \mathcal{F}(\text{RCPSP}[X])$. Montrons que $S \in \mathcal{F}(\text{PARCPSP}[X, \Delta])$.

Soit $\alpha_i^S : \mathbb{R} \to \{0, 1\}$ la fonction d'activation de l'activité $i \in \mathcal{A}$ dans l'ordonnancement S définie par :

$$\alpha_i^S(t) = \begin{cases} 1 & \text{si } S_i \leq t < S_i + p_i \\ 0 & \text{sinon} \end{cases}$$

Remarquons qu'il est possible d'exprimer $D_{i,\ell}(S_i)$ à partir de α_i^S :

$$D_{i,\ell}(S_i) = \int_{(\ell-1)\Delta}^{\ell\Delta} \alpha_i^S(t)\,\mathrm{d}t \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathbb{Z}$$

Puisque S est réalisable pour le RCPSP, S vérifie les contraintes d'antériorité :

$$S_{i_2} - S_{i_1} \geq p_{i_1} \qquad \forall (i_1, i_2) \in E$$

Puisque S est réalisable pour le RCPSP, S vérifie les contraintes de ressources (instantanées) :

$$\sum_{i \in \mathcal{A}} \alpha_i^S(t)\, r_{i,k} \leq b_k \qquad \forall k \in \mathcal{R}\ ,\ \forall t \in \mathbb{R}$$

Par conséquent, pour toute ressource $k \in \mathcal{R}$, dans toute période $[(\ell - 1\Delta), \ell\Delta]$ $(\ell \in \mathbb{Z})$:

$$\sum_{i \in \mathcal{A}} r_{i,k}\, D_{i,\ell}(S_i) = \int_{(\ell-1)\Delta}^{\ell\Delta} \sum_{i \in \mathcal{A}} r_{i,k}\, \alpha_i^S(t)\,\mathrm{d}t \leq b_k\, \Delta$$

S est donc également réalisable pour le PARCPSP. □

La proposition suivante renforce cette notion de relaxation.

Proposition 3.5. *Toute solution réalisable pour le RCPSP est globalement réalisable pour le PARCPSP.*

$$\begin{cases} \forall X \in \mathcal{X} \\ \forall \Delta \in \mathbb{R}_+^* \end{cases} \quad \mathcal{F}(RCPSP[X]) \subseteq \mathcal{GF}(PARCPSP[X, \Delta])$$

Démonstration. Soit $X \in \mathcal{X}$. Soit $\Delta \in \mathbb{R}_+^*$.

D'après la proposition 3.4 :

$$\mathcal{F}(\text{RCPSP}[X]) \subseteq \mathcal{F}(\text{PARCPSP}[X, \Delta])$$

Donc, d'après la proposition 3.2 :

$$\mathcal{GF}(\text{RCPSP}[X]) \subseteq \mathcal{GF}(\text{PARCPSP}[X, \Delta])$$

De plus, pour le RCPSP, les trois concepts de faisabilité sont équivalents.

$$\mathcal{GF}(\text{RCPSP}[X]) = \mathcal{F}(\text{RCPSP}[X]) \qquad \square$$

La proposition 3.5 est plus forte que la proposition 3.4 car, pour le PARCPSP, les trois concepts de faisabilité sont distincts ($\mathcal{GF}(PARCPSP) \subset \mathcal{F}(PARCPSP)$).

De plus, cette proposition montre qu'une solution réalisable du RCPSP n'est pas seulement réalisable, mais réalisable globalement pour le PARCPSP. Cela traduit le fait que de telles solutions sont robustes, en ce sens qu'elles restent réalisables même en étant translatées arbitrairement, et ce, quelle que soit la durée des périodes (Δ).

Les propriétés précédentes mettaient en exergue des liens forts concernant des solutions réalisables (même globalement) ; à l'inverse, la propriété suivante indique l'existence de liens forts concernant des solutions non réalisables (même localement).

Proposition 3.6. *Pour chaque solution non réalisable pour le RCPSP, il existe une valeur seuil de la durée des périodes en-dessous de laquelle cette solution est également non réalisable (même localement) pour le PARCPSP.*

$$\forall X \in \mathcal{X} \quad \forall S \in \overline{\mathcal{F}}(RCPSP[X]) \quad \exists \Delta \in \mathbb{R}_+^* \quad \forall \Delta' \in]0, \Delta] \quad : \quad S \in \overline{\mathcal{LF}}(PARCPSP[X, \Delta'])$$

Démonstration. Soit $X \in \mathcal{X}$. Soit $S \in \overline{\mathcal{F}}(\text{RCPSP}[X])$.

- Si S ne respecte pas les contraintes d'antériorité, S n'est pas réalisable pour le PARCPSP pour toute durée de période $\Delta \in \mathbb{R}_+^*$, pas même localement, puisque translater S n'a pas d'impact vis-à-vis de ces contraintes.
- Si S ne respecte pas les contraintes de ressources (instantanées), il existe une ressource $k \in \mathcal{R}$ et un instant $t \in \mathbb{R}$ tels que :

$$\sum_{i \in \mathcal{A}} r_{i,k}\, \alpha_i^S(t) > b_k$$

Rappelons que tout ordonnancement peut être décomposé en une suite d'intervalles consécutifs dont les bornes coïncident avec des événements de type

début/fin d'exécution d'une activité ; ainsi, pour toute activité $i \in \mathcal{A}$, la fonction α_i^S est constante sur chacun de ces intervalles (fermés à gauche, ouverts à droite).
Cela signifie qu'il existe un intervalle $[t_1, t_2[$ qui contient t et au cours duquel l'ensemble des activités en cours d'exécution est invariant.
Montrons que la propriété est vraie pour $\Delta = \frac{1}{2}(t_2 - t_1)$.
Soit $\Delta' \in]0, \Delta]$. Soit $\ell' = 1 + \left\lceil \frac{t_1}{\Delta'} \right\rceil$.

$$\begin{array}{ll} & \ell' - 2 < \frac{t_1}{\Delta'} \leq \ell' - 1 \\ \Rightarrow & (t_1 \leq (\ell' - 1)\Delta') \wedge (\ell'\Delta' < t_1 + 2\Delta') \\ \Rightarrow & (t_1 \leq (\ell' - 1)\Delta') \wedge (\ell'\Delta' < t_2) \\ \Rightarrow & [(\ell' - 1)\Delta', \ell'\Delta'] \subseteq [t_1, t_2[\end{array}$$

En d'autres termes, la période ℓ', de durée Δ', est incluse dans sa totalité dans l'intervalle $[t_1, t_2[$.
Par conséquent, pour toute activité $i \in \mathcal{A}$, $D_{i,\ell'}(S_i)$ vaut Δ' (si $\alpha_i^S(t) = 1$) ou bien 0 (si $\alpha_i^S(t) = 0$). Autrement dit :

$$\sum_{i \in \mathcal{A}} r_{i,k} \frac{D_{i,\ell'}(S_i)}{\Delta'} > b_k$$

Les contraintes de ressources agrégées sur des périodes de durée Δ' sont donc violées également.
Remarquons que, si S est translaté d'un facteur $\tau \in \mathbb{R}$, alors l'intervalle $[t_1, t_2[$ subit également une translation de facteur τ ; néanmoins, cet intervalle contient toujours une période de durée Δ' dans sa totalité.
Dans tous les cas : $S \in \overline{\mathcal{LF}}(\text{PARCPSP}[X, \Delta])$ □

Le résultat énoncé dans la proposition 3.6 est à nuancer : en effet, le seuil de non-faisabilité dépend de la solution. Par conséquent, il est légitime de se demander si, pour toute instance, il existe un seuil "uniforme" (valable pour toute solution irréalisable du RCPSP) en-dessous duquel le PARCPSP devient équivalent au RCPSP. Formellement, cela revient à déterminer si cette expression est vraie :

$$\forall X \in \mathcal{X} \quad \exists \Delta \in \mathbb{R}_+^* \quad \forall S \in \overline{\mathcal{F}}(\text{RCPSP}[X]) \quad \forall \Delta' \in]0, \Delta] \quad : \quad S \in \overline{\mathcal{LF}}(\text{PARCPSP}[X, \Delta'])$$

Il est possible de simplifier cette expression de la manière suivante :

$$\forall X \in \mathcal{X} \quad \exists \Delta \in \mathbb{R}_+^* \quad \forall \Delta' \in]0, \Delta] \qquad \overline{\mathcal{F}}(\text{RCPSP}[X]) \subseteq \overline{\mathcal{LF}}(\text{PARCPSP}[X, \Delta'])$$

La proposition 3.8 fournit la réponse à cette question : un tel seuil n'existe pas dans le cas général. Pour démontrer ce résultat, il suffit de prouver qu'un tel seuil n'existe pas dans un cas particulier. C'est pourquoi un tel cas est tout d'abord identifié et analysé dans la proposition 3.7.

Proposition 3.7. *Il existe des instances de projet pour lesquelles, pour toute subdivision temporelle uniforme (même avec une durée de périodes arbitrairement petite), le PARCPSP admet des solutions réalisables (même globalement) telles que la valeur du critère est strictement inférieure à la valeur optimale du critère du RCPSP.*

$$\exists X \in \mathcal{X} \quad \forall \Delta \in \mathbb{R}_+^* \quad \exists S \in \mathcal{GF}(PARCPSP[X, \Delta]) \qquad S_{n+1} - S_0 < \text{Opt}(RCPSP[X])$$

Démonstration. Soit $X \in \mathcal{X}$ une instance de projet, paramétrée par quatre entiers naturels non nuls n, p, b et r, définie par (cf. figure 3.1a) :

- $|\mathcal{A}| = n$
- $|\mathcal{R}| = 1$
- $\forall i \in \mathcal{A} \qquad p_i = p$
- $b_1 = b$
- $\forall i \in \mathcal{A} \qquad r_{i,1} = r$
- $E = \emptyset$ (aucune relation d'antériorité)

On fixe les valeurs des paramètres de la manière suivante : $n = 2$, $b = 3$, $r = 2$. Le paramètre p est aussi fixé, mais sa valeur importe peu.

- La somme des consommations des activités ($n \times r = 2 \times 2 = 4$) dépasse la capacité de l'unique ressource ($b = 3$). Pour satisfaire les contraintes de ressource du RCPSP, les fenêtres d'exécution des activités ne peuvent donc pas se chevaucher. Par conséquent, dans toute solution optimale du RCPSP, les $n = 2$ activités sont exécutées l'une après l'autre, sans temps mort (dans n'importe quel ordre ; cf. figure 3.1b).
  D'où : $\mathrm{Opt}(RCPSP[X]) = 2p$
- Soit $\Delta \in \left]0, \frac{p}{4}\right]$. Soit $\varepsilon \in \left]0, \frac{\Delta}{2}\right]$. Soit $S = (0, p - \varepsilon) \in \mathbb{R}^2$ (cf. figure 3.1c). On peut remarquer que la durée du projet définie par S est égale à $2p - \varepsilon < 2p$.
  Montrons que $S \in \mathcal{GF}(\mathrm{PARCPSP}[X, \Delta])$.
  Soit $\tau \in \mathbb{R}$ un décalage temporel (quelconque). Dans la solution $S \oplus \tau$, l'intersection des fenêtres d'exécution des deux activités est égale à l'intervalle $[\tau + p - \varepsilon, \tau + p]$. Puisque $r = 2 < 3 = b$, il suffit de montrer que les contraintes de ressources agrégées sont respectées dans les périodes dont l'intersection avec cet intervalle est non vide. Or, $S \oplus \tau$ vérifie la relation suivante :

  $$D_{1,\ell}(S \oplus \tau) + D_{2,\ell}(S \oplus \tau) \leq \Delta + \varepsilon \qquad \forall \ell \in \mathbb{Z}$$

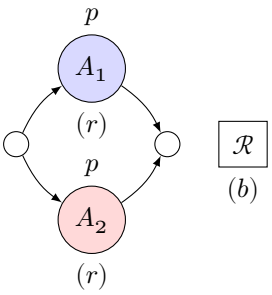
  Par conséquent, la relation suivante est une condition suffisante pour que $S \oplus \tau$ satisfasse les contraintes de ressource agrégées.

  $$r\,(\Delta + \varepsilon) \leq b\,\Delta \quad \Leftrightarrow \quad \varepsilon \leq \Delta \left(\frac{b}{r} - 1\right) = \frac{\Delta}{2}$$

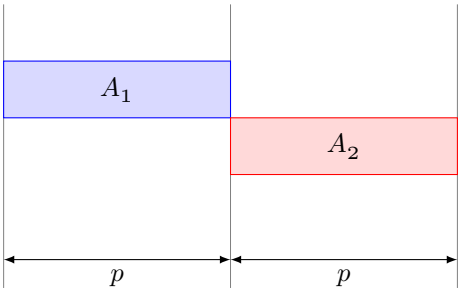
  S est donc globalement réalisable pour une subdivision temporelle uniforme de période $\Delta \leq \frac{p}{4}$. D'après la proposition 3.11 combinée avec la proposition 3.3, S est donc également réalisable pour les subdivisions temporelles uniformes de période $\gamma\Delta$ avec $\gamma \in \mathbb{N}^*$; autrement dit, S est globalement réalisable pour toute subdivision temporelle uniforme. □

Proposition 3.8. *Il existe des instances de projet pour lesquelles, pour toute subdivision temporelle uniforme (même avec une durée de périodes arbitrairement petite), le PARCPSP est une relaxation "au sens strict" du RCPSP, i.e., le PARCPSP admet des solutions optimales qui ne sont pas réalisables pour le RCPSP, et l'optimum du PARCPSP est strictement inférieur à l'optimum du RCPSP.*

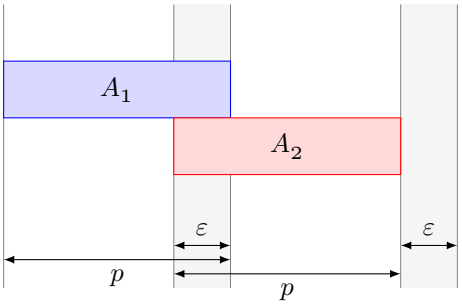
$$\exists X \in \mathcal{X} \quad \forall \Delta \in \mathbb{R}_+^* \qquad \begin{cases} \mathcal{F}^*(PARCPSP[X, \Delta]) \subseteq \overline{\mathcal{F}}(RCPSP[X]) \\ \mathrm{Opt}(PARCPSP[X, \Delta]) < \mathrm{Opt}(RCPSP[X]) \end{cases}$$



(a) Instance de projet



(b) Solution optimale pour le RCPSP



(c) Solution réalisable pour le PARCPSP de durée inférieure

Fig. 3.1 : Chevauchement impossible pour le RCPSP mais possible pour le PARCPSP

Démonstration. Cette propriété est vérifiée par la classe de projets identifiée dans la preuve précédente. □

Le cas particulier qui a été identifié dans la preuve de la proposition 3.7 peut être étendu afin de montrer un résultat négatif sur la qualité de la relaxation du PARCPSP par rapport au RCPSP.

Proposition 3.9. *L'écart absolu entre les fonctions objectifs du RCPSP et du PARCPSP n'est pas borné, même avec une durée de périodes arbitrairement petite, même avec une seule ressource de capacité constante.*

$$\forall E \in \mathbb{R}_+^* \quad \forall \Delta \in \mathbb{R}_+^* \quad \exists X \in \mathcal{X} \qquad \mathrm{Opt}(RCPSP[X]) - \mathrm{Opt}(PARCPSP[X, \Delta]) \geq E$$

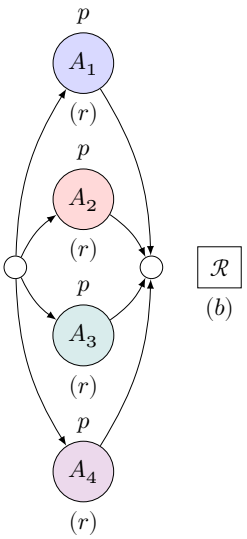
Démonstration. Soit $E \in \mathbb{R}_+^*$. Soit $\Delta \in \mathbb{R}_+^*$. Soit $X \in \mathcal{X}$ un projet paramétré de la même manière que dans la preuve de la proposition 3.7 (cf. figure 3.2a). On fixe les valeurs des paramètres b et r à 2 et 3, respectivement. Le paramètre p est fixé à une valeur supérieure ou égale à 4Δ. Nous allons ici chercher à ajuster le nombre d'activités (paramètre n) afin que le gap absolu entre RCPSP et PARCPSP soit plus grand que E.

- Pour le RCPSP, il est impossible d'exécuter deux activités en parallèle (puisqu'elles sont toutes identiques, et $2r = 4 > 3 = b$).
  D'où : $\mathrm{Opt}(RCPSP[X]) = n\,p$ (cf. figure 3.2b)
- Soit $\varepsilon \in \left]0, \frac{\Delta}{2}\right]$.
  Soit $S \in \mathbb{R}^n$ l'ordonnancement défini par : $\forall i \in \mathcal{A} \quad S_i = (i-1)(p-\varepsilon)$ (cf. figure 3.2c)
  Remarquons que $S_{n+1} - S_0 = n\,p - (n-1)\varepsilon$. De plus, par construction de S, pour toute activité $i \in \mathcal{A}$ sauf la dernière, la fin de la fenêtre d'exécution de i intersecte le début de la fenêtre d'exécution de l'activité $i+1$ sur l'intervalle $[i(p-\varepsilon), (i-1)(p-\varepsilon)+p]$ de largeur ε. Puisque $2\varepsilon \leq \Delta \leq \frac{p}{4}$, la distance entre deux intersections consécutives est supérieure ou égale 2Δ ; cela signifie qu'il existe une période (de durée Δ) incluse en totalité dans l'intervalle (fermé) qui sépare ces deux intersections. Dans cette période, une seule activité est donc en cours d'exécution.
  On en déduit que la condition suffisante $\varepsilon \leq \Delta\left(\frac{b}{r} - 1\right) = \frac{\Delta}{2}$, déjà établie précédemment dans le cas $n = 2$, reste valide pour n quelconque ; ainsi, $S \in \mathcal{GF}(\mathrm{PARCPSP}[X, \Delta])$.
  D'où : $\mathrm{Opt}(RCPSP[X]) - \mathrm{Opt}(PARCPSP[X, \Delta]) \geq (n-1)\varepsilon$
  En conséquence :
  $n \geq 1 + \frac{E}{\varepsilon} \quad \Rightarrow \quad \mathrm{Opt}(RCPSP[X]) - \mathrm{Opt}(PARCPSP[X, \Delta]) \geq E$ □

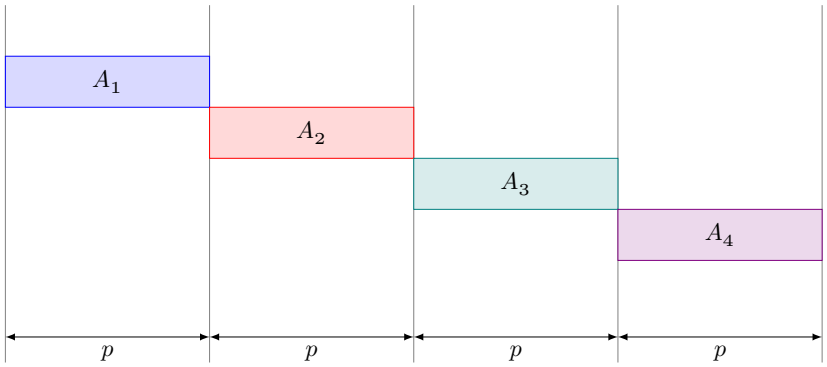
La proposition 3.9 donne également une borne inférieure de l'écart relatif, mais cette borne dépend de la valeur de l'objectif :

$$\frac{\mathrm{Opt}(RCPSP[X])}{\mathrm{Opt}(PARCPSP[X, \Delta])} \geq 1 + \frac{E}{\mathrm{Opt}(PARCPSP[X, \Delta])}$$

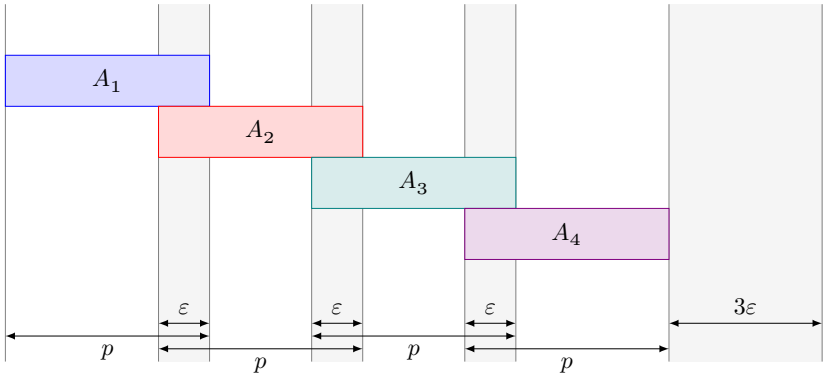
La proposition suivante fournit une borne inférieure de l'écart relatif indépendante de cette valeur, mais dans un cadre plus restreint (nombre de ressources non fixé).



(a) Instance de projet



(b) Solution optimale pour le RCPSP



(c) Solution réalisable pour le PARCPSP de durée inférieure (effet cumulatif)

Fig. 3.2 : Chevauchement impossible pour le RCPSP mais possible pour le PARCPSP – Effet cumulatif

Proposition 3.10. *L'écart relatif entre les fonctions objectifs du RCPSP et du PARCPSP n'est pas borné, même avec une durée de périodes arbitrairement petite, même avec plusieurs ressources de capacités constantes.*

$$\forall E \in \mathbb{R}_+^* \quad \forall \Delta \in \mathbb{R}_+^* \quad \exists X \in \mathcal{X} \qquad \frac{\mathrm{Opt}(RCPSP[X])}{\mathrm{Opt}(PARCPSP[X,\Delta])} \geq E$$

Démonstration. Soit $K_n = (\mathcal{V}_n, \mathcal{E}_n)$ le graphe complet de taille $n \in \mathcal{N}$, i.e. contenant n sommets et tel qu'il existe une arête entre toute paire de sommets distincts.

Soit $E \in \mathbb{R}_+^*$. Soit $\Delta \in \mathbb{R}_+^*$. Soit α et β deux nombres réels tels que $0 < \alpha \leq \beta < 1$.

Soit $X \in \mathcal{X}$ une instance de projet définie par :

- $\mathcal{A} = \mathcal{V}_n$
- $\mathcal{R} = \mathcal{E}_n$
- $\forall i \in \mathcal{A} \qquad p_i = \alpha\Delta$
- $\forall k \in \mathcal{R} \qquad b_k = 2\beta$
- $\forall i \in \mathcal{A} \quad \forall k \in \mathcal{R} \qquad r_{i,k} = 1$ si i est l'une des deux extrémités de l'arête k, 0 sinon
- $E = \emptyset$ (aucune relation d'antériorité)

On peut remarquer que :

- $|\mathcal{A}| = n$
- $|\mathcal{R}| = m = \frac{n(n-1)}{2}$ (puisque $\mathcal{K}_n$ est complet)
- Chaque ressource (arête) est utilisée par exactement deux activités (sommets, i.e. les deux extrémités de l'arête).

Lemme 3.10a (Valeur de l'optimum pour le RCPSP)**.**

$$\mathrm{Opt}(RCPSP[X]) = n\alpha\Delta$$

Démonstration. Sur chaque ressource (utilisée par exactement deux activités), la demande totale ($= 1 + 1 = 2$) dépasse la capacité ($= 2\beta$). Il est donc impossible d'exécuter simultanément deux activités (sommets), puisque celles-ci sont en compétition sur une ressource (l'arête qui les relie). Par conséquent, dans toute solution optimale, les activités sont exécutées les unes à la suite des autres sans temps mort (dans un ordre quelconque). ■

Lemme 3.10b (Valeur de l'optimum pour le PARCPSP)**.**

$$\mathrm{Opt}(PARCPSP[X,\Delta]) = \alpha\Delta$$

Démonstration. Montrons que le vecteur S nul ($\forall i \in \mathcal{A} \quad S_i = 0$) est une solution réalisable.

Puisque toutes les activités commencent à la date $S_0 = 0$, elles terminent également toutes à la même date $S_{n+1} = \alpha\Delta$. De plus, puisqu'il n'y a aucune relation d'antériorité, il suffit de vérifier que S satisfait les contraintes dans la première période $[0, \Delta]$.

Soit $k = (i_1, i_2) \in \mathcal{R}$.

$$\begin{aligned} \textstyle\sum_{i \in \mathcal{A}} r_{i,k} \times d_{i,1}(S) &= r_{i_1,k} \times p_{i_1} + r_{i_2,k} \times p_{i_2} \\ &= 2 \times 1 \times \alpha\Delta \\ &\leq 2\beta \times \Delta \\ &= b_k \times \Delta \end{aligned}$$

D'où : $\text{Opt}(\text{PARCPSP}[X, \Delta]) \leq \alpha\Delta$
De plus : $\forall i \in \mathcal{A} \quad \alpha\Delta = p_i \leq \text{Opt}(\text{PARCPSP}[X, \Delta])$ ■

D'après les lemmes 3.10a et 3.10b, le gap relatif est égal à n, qui peut être choisi arbitrairement grand, en particulier supérieur ou égal à E. □

La question de savoir s'il est possible de borner le gap relatif lorsque le nombre de ressources est limité reste ouverte. Toutefois, une borne inférieure du pire cas est connue ; en effet, pour la classe de projets identifiée dans la preuve de la proposition 3.10, le gap relatif est égal à $\frac{1+\sqrt{1+8m}}{2} = \mathcal{O}(\sqrt{m})$ avec m ressources*.

3.3 Propriétés structurelles du PARCPSP pour des instances définies à partir d'un même projet

Les résultats de la section précédente montrent que, dans le cas général, le PARCPSP n'est pas une bonne approximation du RCPSP. Cette section a pour but d'identifier des relations de relaxation entre des instances du PARCPSP, définies à partir du même projet, mais pour lesquelles les subdivisions temporelles sont différentes, i.e. définies à partir de périodes distinctes. S'il s'avère que de telles relations existent, alors les solutions obtenues en résolvant le problème pour une période donnée peuvent aider à résoudre le problème pour d'autres périodes.

Dans cette section sont présentées les propriétés suivantes :

- existence d'une relation de relaxation entre deux problèmes PARCPSP lorsque la durée des périodes définies dans l'un est une valeur multiple de la durée des périodes définie dans l'autre (proposition 3.11)
- non-existence de telles relations lorsque la condition de multiplicité n'est pas vérifiée (illustrée dans un contre-exemple développé dans la propriété proposition 3.12 pour la faisabilité, et la proposition 3.13 pour l'optimalité)

Proposition 3.11. *Pour tout projet $X \in \mathcal{X}$, si $\Delta_2 \in \mathbb{R}_+^*$ est un multiple de $\Delta_1 \in \mathbb{R}_+^*$, alors le PARCPSP défini avec le projet X et une subdivision temporelle uniforme de période Δ_2 est une relaxation du PARCPSP défini avec le projet X et une subdivision temporelle uniforme de période Δ_1.*

$$\forall X \in \mathcal{X} \quad \forall \Delta \in \mathbb{R}_+^* \quad \forall \gamma \in \mathbb{N}^* \qquad \mathcal{F}(PARCPSP[X, \Delta]) \subseteq \mathcal{F}(PARCPSP[X, \gamma\Delta])$$

Démonstration. Soit $X \in \mathcal{X}$. Soit $\Delta \in \mathbb{R}_+^*$. Soit $\gamma \in \mathbb{N}^*$.

Afin de lever toute ambiguïté, la notation $D_{i,\ell}^{\langle\Delta\rangle}$ (respectivement $D_{i,\ell}^{\langle\gamma\Delta\rangle}$) est utilisée pour désigner la fonction $D_{i,\ell}$ appliquée avec une subdivision temporelle uniforme de durée Δ (respectivement $\gamma\Delta$).

Soit $S \in \mathcal{F}(\text{PARCPSP}[X, \Delta])$: S vérifie donc les contraintes d'antériorité et de ressource agrégées sur des périodes de durée Δ.

$$S_{i_2} - S_{i_1} \geq p_{i_1} \qquad \forall (i_1, i_2) \in E$$

$$\sum_{i \in \mathcal{A}} r_{i,k} \, D_{i,\ell}^{\langle\Delta\rangle}(S_i) \leq b_k \, \Delta \qquad \forall k \in \mathcal{R} \ , \ \forall \ell \in \mathbb{Z}$$

*$\forall n \geq 2 \quad \forall m \geq 1 \qquad m = \frac{n(n-1)}{2} \quad \Leftrightarrow \quad n = \frac{1+\sqrt{1+8m}}{2}$

Pour tout $\ell \in \mathbb{Z}$, la période ℓ définie dans la subdivision temporelle uniforme de durée $\gamma\Delta$ commence à la date $(\ell-1)\gamma\Delta$ et termine à la date $\ell\gamma\Delta$. Dans la subdivision temporelle uniforme de durée Δ, cela correspond à exactement γ périodes (de $(\ell-1)\gamma+1$ à $\ell\gamma$). Par conséquent, pour toute activité $i \in \mathcal{A}$:

$$D_{i,\ell}^{\langle\gamma\Delta\rangle}(S_i) = \sum_{\ell'=(\ell-1)\gamma+1}^{\ell\gamma} D_{i,\ell'}^{\langle\Delta\rangle}(S_i)$$

D'où :

$$\sum_{i\in\mathcal{A}} r_{i,k}\, D_{i,\ell}^{\langle\gamma\Delta\rangle}(S_i) = \sum_{\ell'=(\ell-1)\gamma+1}^{\ell\gamma} \sum_{i\in\mathcal{A}} r_{i,k}\, D_{i,\ell'}^{\langle\Delta\rangle}(S_i) \leq b_k\, \gamma\, \Delta \qquad \forall k \in \mathcal{R}\ ,\ \forall \ell \in \mathbb{Z}$$

S vérifie donc également les contraintes de ressources agrégées sur des périodes de durée $\gamma\Delta$. □

En revanche, si Δ_2 n'est pas un multiple de Δ_1, il n'existe pas *a priori* de relation simple entre les deux problèmes, quels que soient les concepts de faisabilité considérés (faisabilité, faisabilité locale, faisabilité globale). Ce phénomène erratique est étudié par la suite *via* une classe d'instances de projets décrite ci-après.

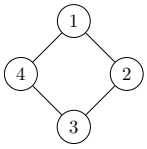
Soit $\mathcal{C}_N = (\mathcal{V}_N, \mathcal{E}_N)$ le N-cyle, i.e. le graphe cycle constitué d'un seul cycle élémentaire de longueur $N \in \mathbb{N}$ (cf. figure 3.3a). Soit p, b et r trois entiers naturels non nuls. À partir de ce graphe et de ces entiers, on définit le projet élémentaire suivant (cf. figure 3.3b).

- $\mathcal{A} = \mathcal{V}_N$
- $\mathcal{R} = \mathcal{E}_N$
- $\forall i \in \mathcal{A} \qquad p_i = p$
- $\forall k \in \mathcal{R} \qquad b_k = b$
- $\forall i \in \mathcal{A} \quad \forall k \in \mathcal{R} \qquad r_{i,k} = r$ si l'activité/sommet i est une extrémité de la ressource/arête k, 0 sinon
- $E = \emptyset$ (aucune relation d'antériorité)

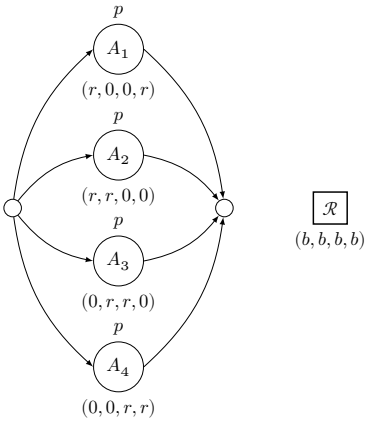
Ce projet élémentaire comporte N activités et N ressources. Afin de construire un projet complet, C clones d'un même projet élémentaire sont juxtaposés de la manière suivante (cf. figure 3.3c).

- Toutes les activités sont conservées ($C \times N$ activités au total).
- Les ressources issues d'une même arête dans $\mathcal{E}_N$ sont considérées comme identiques (N ressources au total).
- Une relation d'antériorité est ajoutée entre une activité du projet élémentaire g (prédecesseur) et une activité du projet élémentaire suivant $g+1$ (successeur) si, et seulement si, ces deux activités correspondent au même sommet dans $\mathcal{V}_N$.

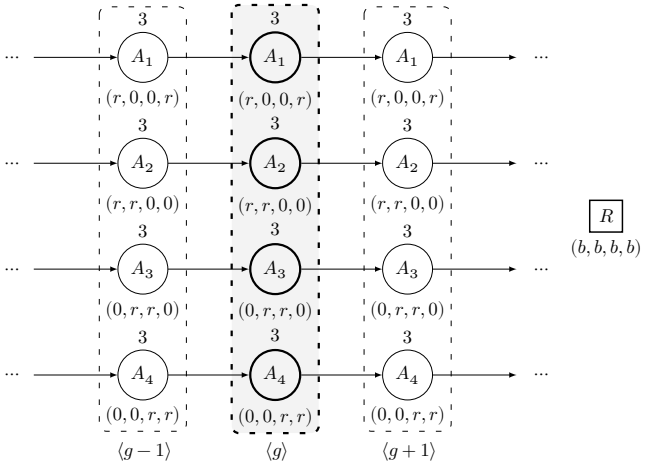
Proposition 3.12. *Il existe des projets pour lesquels une même solution est à la fois réalisable (même globalement) pour une certaine subdivision temporelle uniforme de période $\Delta_1 \in \mathbb{R}_+^*$, et non réalisable (même localement) pour une autre subdivision temporelle uniforme de période $\Delta_2 \in \mathbb{R}_+^*$ ($\Delta_2 > \Delta_1$ ou bien $\Delta_2 < \Delta_1$).*



(a) Graphe cycle

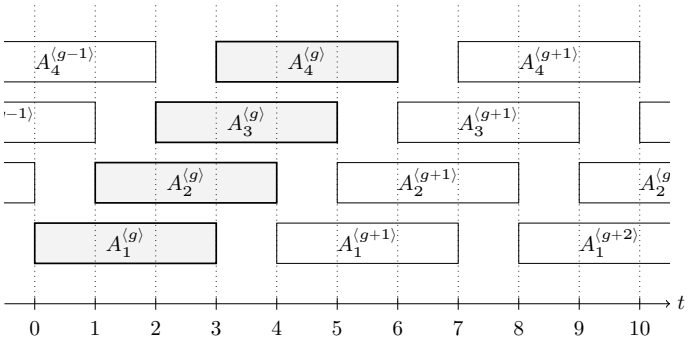


(b) Projet élémentaire

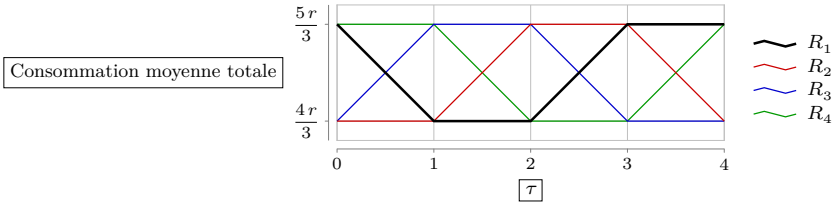


(c) Fusion de projets élémentaires

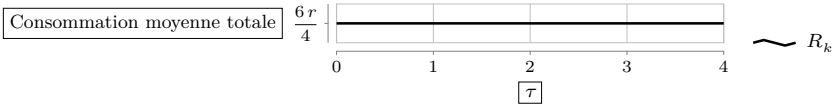
Fig. 3.3 : Construction d'un projet à partir de projets élémentaires identiques issus de graphes cycles



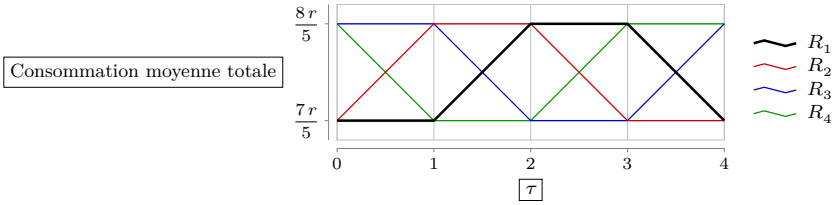
(a) Solution 4-périodique



(b) Consommation moyenne totale dans la première période en fonction du décalage temporel (τ) avec une subdivision uniforme de période $\Delta = 3$



(c) Consommation moyenne totale dans la première période en fonction du décalage temporel (τ) avec une subdivision uniforme de période $\Delta = 4$



(d) Consommation moyenne totale dans la première période en fonction du décalage temporel (τ) avec une subdivision uniforme de période $\Delta = 5$

Fig. 3.4 : Analyse de la consommation moyenne totale en fonction du décalage temporel d'une solution 4-périodique pour différentes subdivisions temporelles uniformes

$$\exists X \in \mathcal{X} \quad \exists \Delta_1 \in \mathbb{R}_+^* \quad \exists \Delta_2 \in \mathbb{R}_+^* \quad : \begin{cases} \Delta_1 \neq \Delta_2 \\ \mathcal{GF}(PARCPSP[X, \Delta_1]) \cap \overline{\mathcal{LF}}(PARCPSP[X, \Delta_2]) \neq \emptyset \end{cases}$$

Démonstration. Montrons que cette propriété est vérifiée par le projet $X^f \in \mathcal{X}$ construit en fusionnant C clones du projet élémentaire $X^e \in \mathcal{X}$ défini par les paramètres $N = 4$, $p = 3$, $b = 3$ et $r = 2$. Dans cette preuve, on suppose que C est arbitrairement grand (les résultats sont valides pour tout $C \geq 4$).

Soit S^e l'ordonnancement des activités de X^e défini par :

$$S_i^e = i - 1 \qquad \forall i \in \{1, \dots, N\}$$

Soit S^f l'ordonnancement des activités de X^f obtenu de la manière suivante : les activités du $g^{\text{ième}}$ clone de X^e ($g \in \mathbb{Z}$) sont exécutées suivant l'ordonnancement $S^e \oplus 4g$ (cf. figure 3.4a).

Par construction de X^f et S^f, l'ordonnancement obtenu est 4-périodique ; il en est de même pour la consommation instantanée sur toutes les ressources, et donc pour la consommation moyenne selon une subdivision temporelle uniforme de période quelconque.

Les figures 3.4b à 3.4d représentent la consommation moyenne de l'ordonnancement $S^f \oplus \tau$ dans la période $\ell = 1$, i.e. l'intervalle de temps $[0, \Delta]$, en fonction du décalage temporel $\tau \in [0, 4[$, pour trois valeurs distinctes des durées des périodes ($\Delta \in \{3, 4, 5\}$).

- $\Delta = 3$
  Pour toute valeur de τ, la consommation moyenne totale atteint $\frac{5r}{3}$ sur (au moins) une ressource.
  Puisque $\frac{5r}{3} = \frac{10}{3} > 3 = b$, les contraintes de ressource agrégées ne sont pas respectées.
  Par conséquent : $S \in \overline{\mathcal{LF}}(PARCPSP[X^f, 3])$
- $\Delta = 4$
  Pour toute valeur de τ, la consommation moyenne totale est toujours égale à $\frac{6r}{4}$ sur toutes les ressources.
  Puisque $\frac{6r}{4} = \frac{12}{4} \leq 3 = b$, les contraintes de ressource agrégées sont respectées.
  Par conséquent : $S \in \mathcal{GF}(PARCPSP[X^f, 4])$
- $\Delta = 5$
  Pour toute valeur de τ, la consommation moyenne totale atteint $\frac{7r}{5}$ sur (au moins) une ressource.
  Puisque $\frac{8r}{5} = \frac{16}{5} > 3 = b$, les contraintes de ressource agrégées ne sont pas respectées.
  Par conséquent : $S \in \overline{\mathcal{LF}}(PARCPSP[X^f, 5])$

La proposition a donc bien été établie, dans les deux cas $\Delta_1 > \Delta_2$ ($\Delta_1 = 4$ et $\Delta_2 = 3$) et $\Delta_1 < \Delta_2$ ($\Delta_1 = 4$ et $\Delta_2 = 5$). □

Proposition 3.13. *Il existe des projets pour lesquels la valeur optimale de la fonction objectif n'est pas une fonction monotone de la durée des périodes.*

Démonstration. Soit $X^f \in \mathcal{X}$ le projet issu de la fusion de $C = 3$ clones du projet élémentaire $X^e \in \mathcal{X}$ défini avec les mêmes paramètres que précédemment ($N = 4$, $p = 3$, $b = 3$ et $r = 2$).

On cherche à résoudre les problèmes PARCPSP$[X^f, \Delta]$, avec $\Delta \in \{3, 4, 5\}$ de manière optimale. Les solutions optimales présentées ci-dessous sont celles retournées par un solveur. Il s'avère que, pour ces trois valeurs de Δ, les solutions renvoyées ont la même structure : toutes les activités issues d'un même projet élémentaire sont exécutées simultanément.

- $\Delta = 3$
  1. Projet élémentaire 1 : début à $t = 0.75$, fin à $t = 3.75$
  2. Projet élémentaire 2 : début à $t = 4.5$, fin à $t = 7.5$
  3. Projet élémentaire 3 : début à $t = 8.25$, fin à $t = 11.25$

  La durée de projet optimale est donc égale à 10.5.
- $\Delta = 4$
  1. Projet élémentaire 1 : début à $t = 1$, fin à $t = 4$
  2. Projet élémentaire 2 : début à $t = 5$, fin à $t = 8$
  3. Projet élémentaire 3 : début à $t = 8$, fin à $t = 11$

  La durée de projet optimale est donc égale à 10.
- $\Delta = 5$
  1. Projet élémentaire 1 : début à $t = 1.25$, fin à $t = 4.25$
  2. Projet élémentaire 2 : début à $t = 4.5$, fin à $t = 7.5$
  3. Projet élémentaire 3 : début à $t = 8.5$, fin à $t = 11.5$

  La durée de projet optimale est donc égale à 10.25.

On peut remarquer que l'optimum pour $\Delta = 4$ (10) est strictement inférieur à l'optimum pour $\Delta = 3$ (10.5) et pour $\Delta = 5$ (10.25). La proposition est donc bien établie. □

La proposition 3.13 implique notamment que, pour un projet donné, la valeur de l'optimum du PARCPSP n'est pas une fonction décroissante de la durée des périodes dans le cas général.

3.4 Algorithme polynomial pour déterminer si une solution est réalisable, réalisable localement et réalisable globalement

Dans les sections précédentes, de nouveaux concepts ont été introduits afin de décrire de manière précise la faisabilité d'un ordonnancement. Ces nouveaux concepts ont permis de formaliser des propriétés liées à l'impact de la variation de la durée des périodes sur la faisabilité d'un ordonnancement.

Dans le chapitre précédent (section complexité), il a été montré que le PARCPSP appartient à la classe des problèmes non-déterministes polynomiaux (**NP**). Autrement dit, il existe un algorithme polynomial permettant de tester la faisabilité d'un ordonnancement.

Dans cette section, un algorithme polynomial permettant de tester non seulement la faisabilité mais aussi la faisabilité locale et la faisabilité globale d'un ordonnancement $S \in \mathbb{R}^n$ est proposé. Plus précisément, cet algorithme permet de déterminer l'ensemble des décalages temporels $\tau \in \mathbb{R}$ tels que l'ordonnancement $S \oplus \tau$ est réalisable.

Rappelons que la satisfaction des contraintes d'antériorité peut être vérifiée en temps polynomial (problème central de l'ordonnancement) ; c'est pourquoi nous

nous focalisons exclusivement sur les contraintes de ressources agrégées par période dans la suite de cette section.

3.4.1 Recherche de l'ensemble des décalages temporels pour lesquels un ordonnancement donné est réalisable

Pour le PARCPSP, les contraintes de ressources agrégées par périodes s'écrivent de la manière suivante :

$$\sum_{i \in \mathcal{A}} r_{i,k}\, D_{i,\ell}(S_i) \leq b_k\, \Delta \qquad \forall k \in \mathcal{R}\ ,\ \forall \ell \in \mathbb{Z}$$

Pour chaque ressource $k \in \mathcal{R}$, soit f_k la fonction qui, à un offset $\tau \in \mathbb{R}$, associe la valeur du membre de gauche de la contrainte sur la ressource k agrégée dans la première période ($\ell = 1$), i.e. l'intervalle $[0, \Delta]$, pour la solution $S \oplus \tau$:

$$f_k(\tau) = \sum_{i \in \mathcal{A}} r_{i,k}\, D_{i,1}(S_i + \tau)$$

Pour tout $\ell \in \mathbb{Z}$, $f_k(\ell\Delta)$ peut être interprété de deux manières différentes.

- Par définition, c'est la valeur du membre de gauche pour la solution $S \oplus \ell\Delta$ dans la contrainte sur la ressource k agrégée sur la période 1 i.e. l'intervalle $[0, \Delta]$.
- C'est aussi la valeur du membre de gauche pour la solution S dans la contrainte sur la ressource k agrégée sur une autre période, en l'occurrence la période $1 - \ell$ i.e. l'intervalle $[-\ell\Delta, (1-\ell)\Delta]$.

En effet, pour toute activité $i \in \mathcal{A}$:

$$\begin{array}{ll} & D_{i,1}(S_i + \ell\Delta) \\ = & \max(\ 0\ ,\ \min(\Delta, S_i + \ell\Delta + p_i) - \max(0, S_i + \ell\Delta)\ ) \\ = & \max(\ 0\ ,\ \min((1-\ell)\Delta, S_i + p_i) + \cancel{\ell\Delta} - \max(-\ell\Delta, S_i) - \cancel{\ell\Delta}\ ) \\ = & D_{i,1-\ell}(S_i) \end{array}$$

Soit $S_0 = \min_{1 \leq i \leq n}(S_i)$ (date de début d'exécution de l'ordonnancement) et $S_{n+1} = \max_{1 \leq i \leq n}(S_i + p_i)$ (date de fin d'exécution de l'ordonnancement) .

- $f_k(\tau)$ est nul si $S_0 + \tau \geq \Delta$ (après la première période) i.e. $\tau \geq \tau_{\max} = \Delta - S_0$
- $f_k(\tau)$ est nul si $S_{n+1} + \tau \leq 0$ (avant la première période) i.e. $\tau \leq \tau_{\min} = -S_{n+1}$

Remarquons que chaque fonction f_k est une combinaison linéaire d'un nombre polynomial de fonctions continues, linéaires par morceaux, avec un nombre constant de morceaux : en effet, les fonctions $D_{i,\ell}$ sont continues, linéaires par morceaux, avec un nombre constant de morceaux : 4 morceaux si $p_i = \Delta$, 5 morceaux sinon. Chaque fonction f_k est donc continue, linéaire par morceaux, avec un nombre polynomial de morceaux. Ainsi, pour toute borne supérieure $u \in \mathbb{R}_+$, il est possible de calculer en temps polynomial l'ensemble (domaine) $D_k(u)$ défini par :

$$D_k(u) = \{\tau \in [\tau_{\min}, \tau_{\max}]\ :\ f_k(\tau) \leq u\}$$

Puisque chaque fonction f_k est linéaire par morceaux, l'ensemble des τ solutions de l'équation $f_k(\tau) \leq u$ sur un morceau donné est soit un ensemble vide, soit un intervalle fermé (éventuellement réduit à un seul point). De plus, le nombre

de morceaux est polynomial. Ainsi, $D_k(u)$ est une union d'un nombre polynomial d'intervalles fermés (éventuellement réduits à un seul point).

Par conséquent, pour toutes bornes supérieures réelles positives $u_1, \dots, u_m$, l'ensemble (domaine) suivant peut-être calculé en temps polynomial.

$$D(u_1, \dots, u_m) = \bigcap_{k=1}^{m} D_k(u_k)$$

Soit $\mathcal{D} = D(b_1\Delta, \dots, b_m\Delta)$: $\mathcal{D}$ est l'union des intervalles fermés contenant les offsets $\tau \in [\tau_{\min}, \tau_{\max}]$ tels que $S \oplus \tau$ satisfait les contraintes de ressources agrégées sur toutes les ressources dans la période $[0, \Delta]$ ($\ell = 1$).

Soit $\overline{\mathcal{D}}$ l'ensemble $[\tau_{\min}, \tau_{\max}]$ privé de $\mathcal{D}$: $\overline{\mathcal{D}}$ est l'union des intervalles ouverts contenant les offsets $\tau \in [\tau_{\min}, \tau_{\max}]$ tels que $S \oplus \tau$ viole les contraintes de ressources agrégées sur au moins une ressource dans la période $[0, \Delta]$ ($\ell = 1$).

1. S est réalisable *si et seulement si* : $\mathcal{D} \supseteq \Delta\mathbb{Z}$[†]
   Ainsi, pour toute ressource k, pour tout $\ell \in \mathbb{Z}$: $\quad f_k(\ell\Delta) \leq b_k\,\Delta$
2. S est réalisable localement *si et seulement si* : $\exists \tau^* \in [0, \Delta[\quad \mathcal{D} \supseteq \tau^* + \Delta\mathbb{Z}$[‡]
   Ainsi, pour toute ressource k, pour tout $\ell \in \mathbb{Z}$: $\quad f_k(\tau^* + \ell\Delta) \leq b_k\,\Delta$
3. S est réalisable globalement *si et seulement si* : $\overline{\mathcal{D}} = \emptyset$
   Ainsi, pour toute ressource k, pour tout $\tau \in \mathbb{R}$: $\quad f_k(\tau) \leq b_k\,\Delta$

3.4.2 Agrégation des résultats sur une période

Les conditions précédentes sont basées sur des relations d'inclusion entre ensembles. En raisonnant "modulo Δ" (agrégation des résultats sur une seule période), il est possible de trouver des conditions équivalentes à celles-ci, plus simples (basées sur l'appartenance d'un élément à un ensemble).

Soit ($a \bmod b$) le reste de la division euclidienne de a par b étendue aux réels($a \in \mathbb{R}$ et $b \in \mathbb{R}_+^*$) défini par :

$$a \bmod b = a - \left\lfloor \frac{a}{b} \right\rfloor b$$

Soit $\mathrm{Mod}_b(X)$ l'image de $X \subseteq \mathbb{R}$ par ($\bullet \bmod b$) où $b \in \mathbb{R}_+^*$, i.e. :

$$\mathrm{Mod}_b(X) = \left\{ y \in [0, b[\ : \ \exists x \in X \quad y = x \bmod b \right\}$$

Les deux propriétés suivantes assurent que la complexité de l'algorithme reste polynomiale.

1. Si X est un intervalle, alors $\mathrm{Mod}_b(X)$ peut-être calculé en temps constant.

$$\mathrm{Mod}_b([x, y]) = \begin{cases} [0, b[& \text{si } y - x \geq b \\ [x \bmod b, y \bmod b] & \text{si } y - x < b \text{ et } \lfloor \frac{x}{b} \rfloor = \lfloor \frac{y}{b} \rfloor \\ [0, y \bmod b] \cup [x \bmod b, b[& \text{si } y - x < b \text{ et } \lfloor \frac{x}{b} \rfloor < \lfloor \frac{y}{b} \rfloor \end{cases}$$

[†] $\tau \in \Delta\mathbb{Z} \quad \Leftrightarrow \quad \exists \ell \in \mathbb{Z} \quad \tau = \ell\Delta$
[‡] $\tau \in \tau^* + \Delta\mathbb{Z} \quad \Leftrightarrow \quad \exists \ell \in \mathbb{Z} \quad \tau = \tau^* + \ell\Delta$

2. Si X est une union d'intervalles, alors $\mathrm{Mod}_b(X)$ peut-être calculé en temps linéaire par rapport au nombre d'intervalles, grâce à la propriété suivante (endomorphisme).

$$\mathrm{Mod}_b([x_1, y_1] \cup [x_2, y_2]) = \mathrm{Mod}_b([x_1, y_1]) \cup \mathrm{Mod}_b([x_2, y_2])$$

Soit $\overline{T} = \mathrm{Mod}_\Delta(\overline{\mathcal{D}})$: $\overline{T}$ est l'ensemble des offsets $\tau \in [0, \Delta[$ tels que $S \oplus \tau$ n'est pas réalisable.

Soit T l'ensemble $[0, \Delta[$ privé de $\overline{T}$: T est l'ensemble des offsets $\tau \in [0, \Delta[$ tels que $S \oplus \tau$ est réalisable.

1. S est réalisable *si et seulement si* : $T \ni 0$
2. S est réalisable localement *si et seulement si* : $T \neq \emptyset$
3. S est réalisable globalement *si et seulement si* : $\overline{T} = \emptyset$

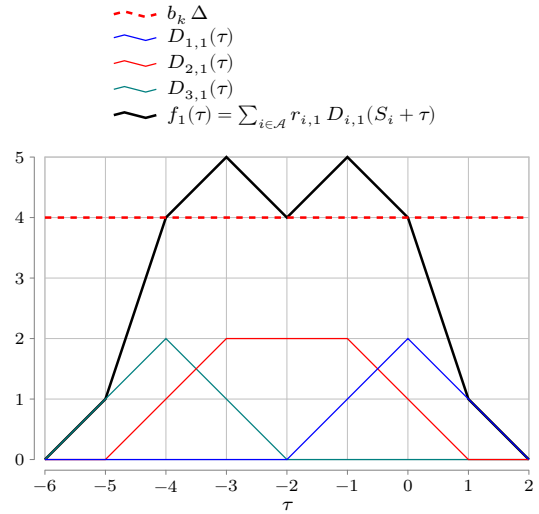
Remarque. Ces conditions sont bien cohérentes avec la proposition 3.1 : la faisabilité globale implique la faisabilité, qui elle-même implique la faisabilité locale.

3.4.3 Exemple

L'algorithme qui découle du raisonnement présenté précédemment est illustré sur le deuxième exemple du chapitre précédent (cf. figure 2.9 page 31). On s'intéresse à l'impact d'un décalage temporel en terme de faisabilité du même ordonnancement : $S = (0, 1, 4)$. Les mêmes subdivisions temporelles sont considérées :

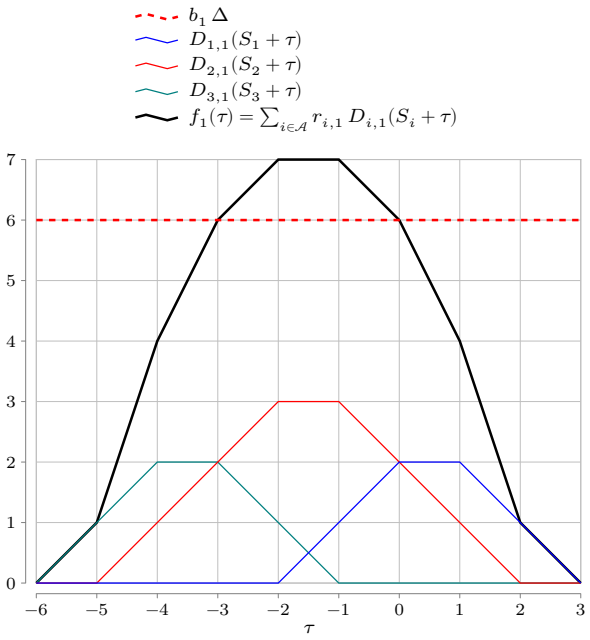
- $\Delta = 2$: cf. figure 3.5 page 65
- $\Delta = 3$: cf. figure 3.6 page 66
- $\Delta = 4$: cf. figure 3.7 page 67
- $\Delta = 5$: cf. figure 3.8 page 68

Pour chaque valeur de Δ, l'ensemble T trouvé est bien cohérent avec les résultats proposés dans le tableau 2.1 page 31 (décalages temporels τ entiers).



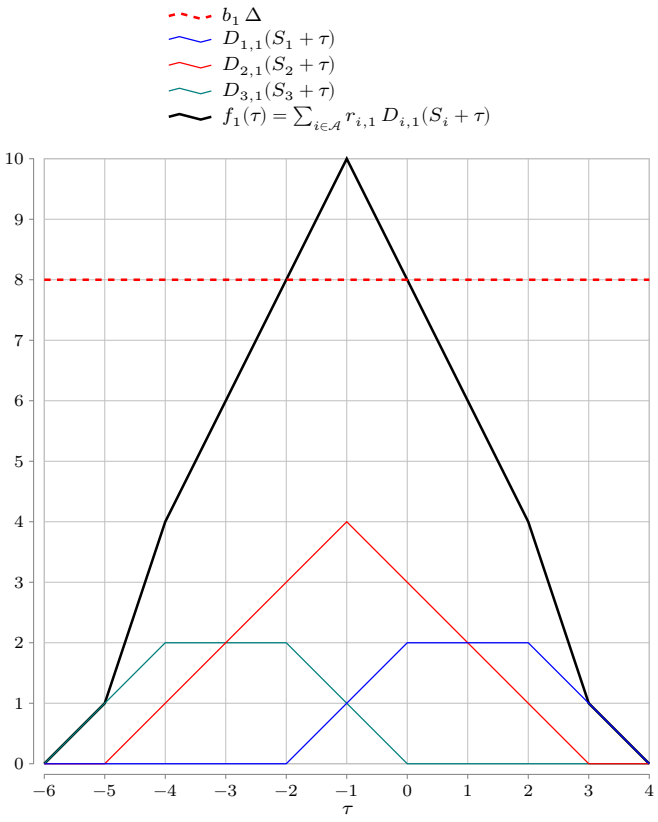
$$\mathcal{D} = [-6, -4] \cup \{-2\} \cup [0, 2]$$
$$\overline{\mathcal{D}} =]-4, -2[\cup]-2, 0[$$
$$\overline{T} =]0, 2[$$
$$T = \{0\}$$

Fig. 3.5 : Analyse de la sensibilité d'un ordonnancement à un décalage temporel en terme de faisabilité – Illustration pour le cas $\Delta = 2$



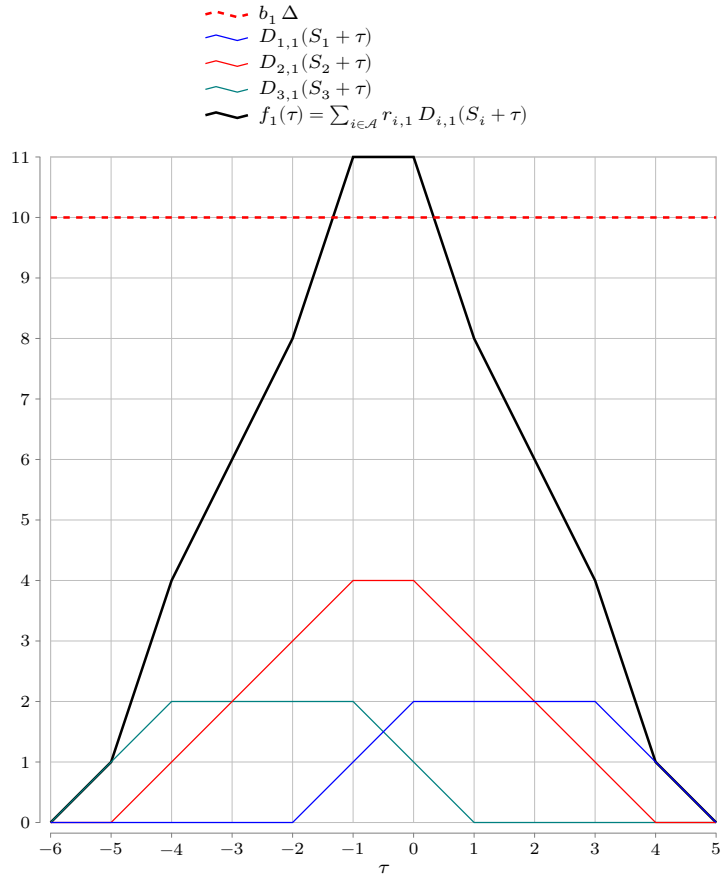
$$\mathcal{D} = [-6,-3] \cup [0,3]$$
$$\overline{\mathcal{D}} =]-3,0[$$
$$\overline{T} =]0,3[$$
$$T = \{0\}$$

Fig. 3.6 : Analyse de la sensibilité d'un ordonnancement à un décalage temporel en terme de faisabilité – Illustration pour le cas $\Delta = 3$



$$\mathcal{D} = [-6,-2] \cup [0,4]$$
$$\overline{\mathcal{D}} =]-2,0[$$
$$\overline{T} =]2,4[$$
$$T = [0,2]$$

Fig. 3.7 : Analyse de la sensibilité d'un ordonnancement à un décalage temporel en terme de faisabilité – Illustration pour le cas $\Delta = 4$



$$\begin{aligned}
\mathcal{D} &= \left[-6, -\tfrac{4}{3}\right] \cup \left[\tfrac{1}{3}, 5\right] \\
\overline{\mathcal{D}} &= \left]-\tfrac{4}{3}, \tfrac{1}{3}\right[\\
\overline{T} &= \left[0, \tfrac{1}{3}\right[\cup \left]\tfrac{11}{3}, 5\right[\\
T &= \left[\tfrac{1}{3}, \tfrac{11}{3}\right]
\end{aligned}$$

Fig. 3.8 : Analyse de la sensibilité d'un ordonnancement à un décalage temporel en terme de faisabilité – Illustration pour le cas $\Delta = 5$

Chapitre 4

PARCPSP : méthodes de résolution exactes basées sur la programmation linéaire en nombres entiers

Dans ce chapitre, plusieurs formulations de programmation linéaire en nombres entiers sont proposées pour le PARCPSP. De par la nature des contraintes de ressources agrégées, les modèles reposent sur l'expression de la longueur de l'intersection de l'intervalle d'exécution des activités et des périodes. C'est pourquoi une analyse de cette fonction est proposée en préambule : il est montré que cette fonction est une somme de fonctions linéaires par morceaux continues et monotones. Des bornes sur le nombre de périodes d'exécution d'une activité sont également données. Des modèles à dates de début continues et comprenant des variables binaires indexées par les périodes sont ensuite proposés sur la base de cette analyse. Deux formulations sont présentées, et il est établi que la deuxième est strictement plus forte que la première en termes de relaxation continue. Des évaluations expérimentales sont fournies dans le chapitre 5. Enfin, nous proposons un modèle compact, au sens où le nombre de variables et de contraintes ne dépend pas du nombre de périodes.

Les formulations présentées dans ce chapitre et leurs relations ont fait l'objet de plusieurs publications dans des conférences et une revue internationale (Morin et al. 2016a,b ; Morin, Artigues et Haït 2017a,b ; Morin et al. 2018b).

4.1 Exploitation de la structure du problème pour caractériser la fenêtre d'exécution d'une activité

Les modèles mathématiques présentés dans ce chapitre reposent sur l'expression, pour chaque activité et chaque période, de la longueur de l'intersection de la fenêtre d'exécution de l'activité avec la période. Ainsi, dans la section 4.1.1, nous effectuons une analyse structurelle de cette longueur qui peut être définie comme une fonction de la date de début de la tâche qui aboutit à une décomposition utile pour la programmation linéaire. Par ailleurs, le nombre de variables binaires nécessaires dépend du nombre de périodes intersectées par la fenêtre d'exécution. Dans

la section 4.1.2 nous donnons un encadrement de ce nombre.

4.1.1 Analyse de la fonction d'évaluation de la durée d'exécution d'une activité dans une période

Définition (rappel)

Dans le chapitre 2, la fonction $D_{i,\ell}$ a été définie de la manière suivante : à toute date $t \in \mathbb{R}$, elle associe la longueur de l'intersection des intervalles $[t, t+p_i]$ et $[(\ell-1)\Delta, \ell\Delta]$. Ainsi, $D_{i,\ell}(t)$ peut être interprété comme la durée d'exécution de l'activité $i \in \mathcal{A}$ dans la période $\ell \in \mathbb{Z}$ lorsque l'activité i démarre à la date $t \in \mathbb{R}$.

$$D_{i,\ell} = \left\{ \begin{array}{ccl} \mathbb{R} & \rightarrow & [0, \Delta] \\ t & \mapsto & \max\left(0 \ , \ \min\left(t + p_i, \ell\Delta\right) - \max\left(t, (\ell-1)\Delta\right) \right) \end{array} \right. \tag{4.1}$$

Une représentation graphique de la fonction $D_{i,\ell}$ est proposée dans la figure 4.1a page 71.

Décomposition en somme de fonctions continues, linéaires par morceaux, monotones

Considérons à présent deux autres fonctions, définies de manière très similaire.

Soit $\Lambda_{i,\ell}$ la fonction qui, à toute date $t \in \mathbb{R}$, associe la longueur de l'intersection des intervalles $]-\infty, t]$ et $[(\ell-1)\Delta, \ell\Delta]$. Ainsi, $\Lambda_{i,\ell}(t)$ peut être interprété comme la durée *avant* l'exécution de l'activité $i \in \mathcal{A}$ dans la période $\ell \in \mathbb{Z}$ lorsque l'activité i démarre à la date t.

$$\Lambda_{i,\ell} = \left\{ \begin{array}{ccl} \mathbb{R} & \rightarrow & [0, \Delta] \\ t & \mapsto & \max\left(0 \ , \ \min\left(\Delta, t - (\ell-1)\Delta\right) \right) \end{array} \right.$$

Soit $\mathrm{M}_{i,\ell}$ la fonction qui, à toute date $t \in \mathbb{R}$, associe la longueur de l'intersection des intervalles $[t+p_i, +\infty[$ et $[(\ell-1)\Delta, \ell\Delta]$. Ainsi, $\mathrm{M}_{i,\ell}(t)$ peut être interprété comme la durée *après* l'exécution de l'activité $i \in \mathcal{A}$ dans la période $\ell \in \mathbb{Z}$ lorsque l'activité i démarre à la date t.

$$\mathrm{M}_{i,\ell} = \left\{ \begin{array}{ccl} \mathbb{R} & \rightarrow & [0, \Delta] \\ t & \mapsto & \max\left(0 \ , \ \min\left(\Delta, \ell\Delta - (t + p_i)\right) \right) \end{array} \right.$$

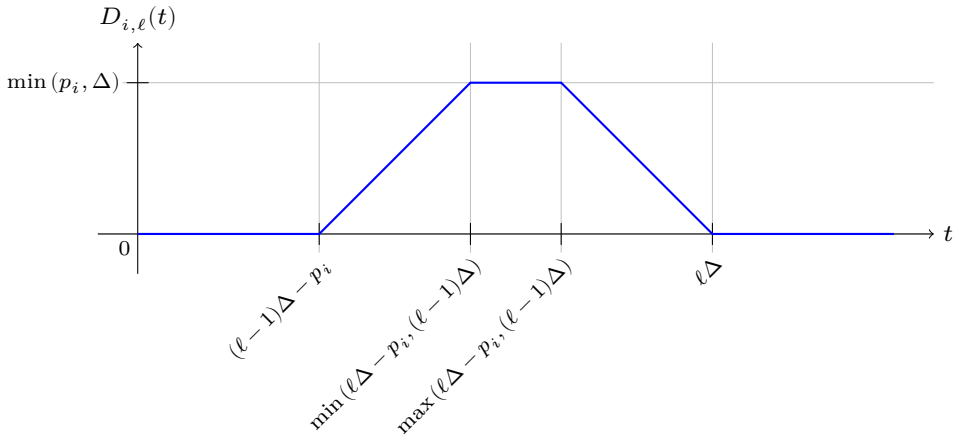
Des représentations graphiques des fonctions $\Lambda_{i,\ell}$ et $\mathrm{M}_{i,\ell}$ sont proposées dans les figures 4.1b et 4.1c page 71.

Partitionnement d'une période

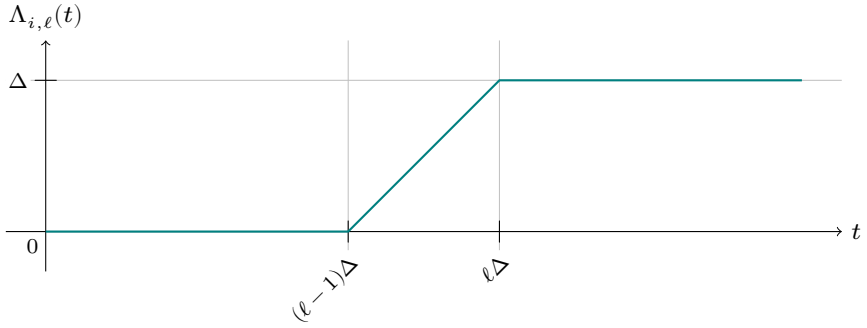
À partir des fonctions introduites précédemment, il est possible d'écrire un invariant, valable pour toute activité $i \in \mathcal{A}$ dans toute période $\ell \in \mathbb{Z}$.

Proposition 4.1 (Partitionnement d'une période)**.**

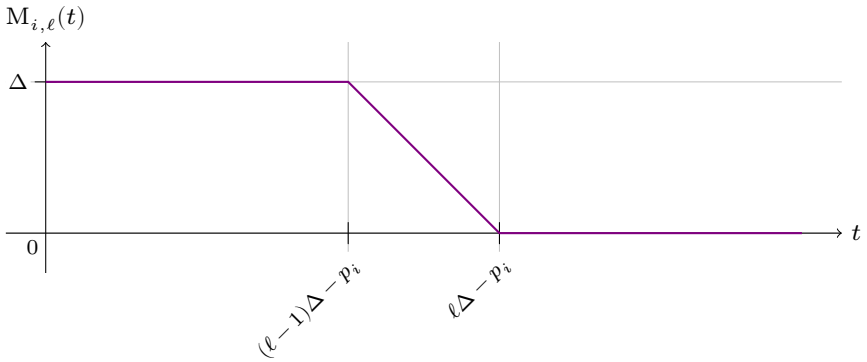
$$\left\{ \begin{array}{l} \forall i \in \mathcal{A} \\ \forall \ell \in \mathbb{Z} \\ \forall t \in \mathbb{R} \end{array} \right. \qquad \Lambda_{i,\ell}(t) + D_{i,\ell}(t) + \mathrm{M}_{i,\ell}(t) = \Delta$$



(a) $D_{i,\ell}$

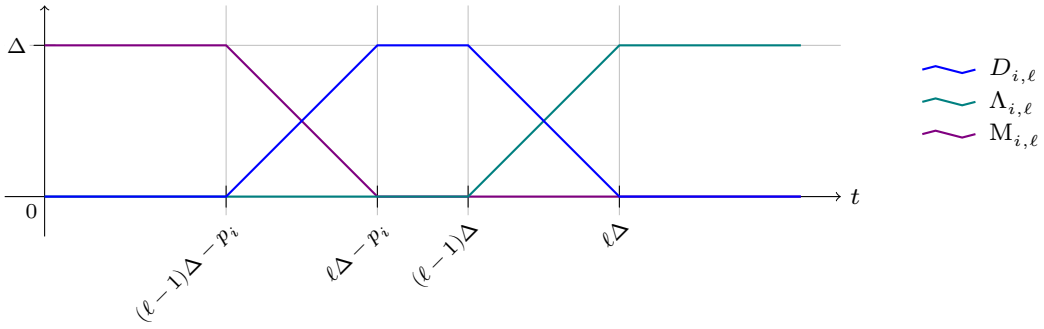


(b) $\Lambda_{i,\ell}$

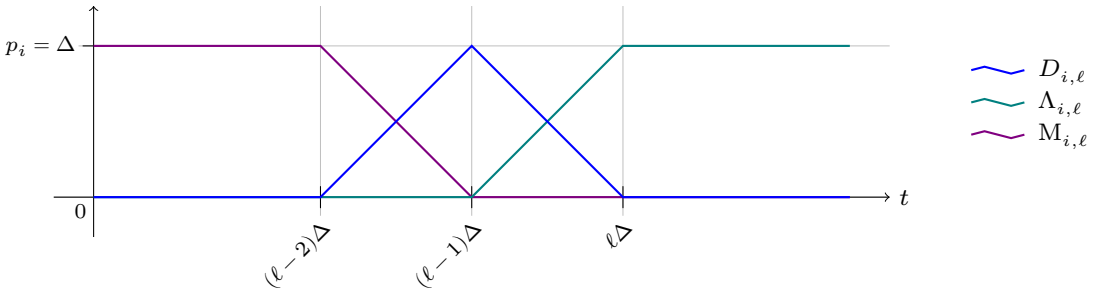


(c) $\mathrm{M}_{i,\ell}$

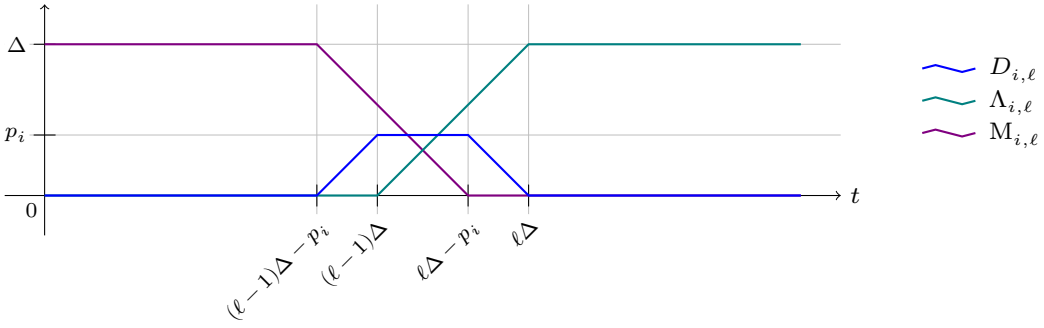
Fig. 4.1 : Fonctions linéaires par morceaux $D_{i,\ell}$, $\Lambda_{i,\ell}$ et $\mathrm{M}_{i,\ell}$



(a) $p_i > \Delta$



(b) $p_i = \Delta$



(c) $p_i < \Delta$

Fig. 4.2 : Somme invariante : $\Lambda_{i,\ell}(t) + D_{i,\ell}(t) + \mathrm{M}_{i,\ell}(t) = \Delta$

Démonstration. Soit $\mathrm{Ind}_\ell : \mathbb{R} \to \{0,1\}$ l'indicatrice de la période $\ell \in \mathbb{Z}$, i.e. $\mathrm{Ind}_\ell(t) = 1$ si $t \in [(\ell-1)\Delta, \ell\Delta]$, 0 sinon. Remarquons que Ind_ℓ est l'indicatrice d'un intervalle de longueur Δ.

$$\begin{aligned}
&\Lambda_{i,\ell}(t) + D_{i,\ell}(t) + \mathrm{M}_{i,\ell}(t) \\
&= \int_{-\infty}^{t} \mathrm{Ind}_\ell(u)\,\mathrm{d}u + \int_{t}^{t+p_i} \mathrm{Ind}_\ell(u)\,\mathrm{d}u + \int_{t+p_i}^{+\infty} \mathrm{Ind}_\ell(u)\,\mathrm{d}u \\
&= \int_{-\infty}^{+\infty} \mathrm{Ind}_\ell(u)\,\mathrm{d}u = \Delta
\end{aligned}$$

□

Cette propriété est illustrée dans la figure 4.2 page 72.

Autrement dit, il est possible d'exprimer $D_{i,\ell}$ comme une somme de fonctions continues, linéaires par morceaux et monotones.

$$D_{i,\ell}(t) = \Delta - \Lambda_{i,\ell}(t) - \mathrm{M}_{i,\ell}(t)$$

Dans les modèles mathématiques présentés dans la suite de ce chapitre, il sera nécessaire de calculer les valeurs $D_{i,\ell}(S_i)$ afin d'évaluer la consommation moyenne dans chaque période, qui intervient dans la contrainte de ressource agrégée. Or, pour une activité $i \in \mathcal{A}$ donnée, la fonction $\ell \mapsto D_{i,\ell}(S_i)$ n'est pas monotone, alors que les fonctions $\ell \mapsto \Lambda_{i,\ell}(S_i)$ et $\ell \mapsto \mathrm{M}_{i,\ell}(S_i)$ le sont (décroissante et croissante, respectivement). L'utilisation de ces fonctions auxiliaires permet d'améliorer la relaxation linéaire, notamment en désagrégeant les contraintes d'antériorité.

4.1.2 Nombre de périodes intersectées par une fenêtre d'exécution

Soit $i \in \mathcal{A}$ une activité. Soit $t \in \mathbb{R}$ une date de début potentiel pour cette activité. Soit ℓ_i^S (respectivement ℓ_i^C) l'indice d'une période qui contient t (respectivement $t + p_i$). Est-il possible de borner l'écart entre ℓ_i^S et ℓ_i^C ?

Proposition 4.2 (Encadrement du nombre de périodes intersectées par une fenêtre d'exécution)**.** *Il existe toujours une configuration de début/fin* (ℓ_i^S, ℓ_i^C) *telle que :*

$$\left\lfloor \frac{p_i}{\Delta} \right\rfloor \leq \ell_i^C - \ell_i^S \leq \left\lceil \frac{p_i}{\Delta} \right\rceil$$

En particulier :

- *Dans le cas où* $p_i \bmod \Delta \neq 0$*, cet encadrement autorise seulement deux valeurs consécutives pour* $\ell_i^C - \ell_i^S$*.*
- *Dans le cas où* $p_i \bmod \Delta = 0$*, cet encadrement n'autorise qu'une seule valeur pour* $\ell_i^C - \ell_i^S$*.*

Démonstration. Une énumération de l'ensemble des cas possibles permet d'affirmer que cet encadrement est valide.

La figure 4.3 page 75 propose une illustration pour chacun des quatre cas possibles lorsque $p_i \bmod \Delta \neq 0$, i.e. p_i n'est pas un multiple de Δ (dans la figure, le ratio $\frac{p_i}{\Delta}$ vaut $3,5$). En fonction des cas, il existe une ou deux configurations de début/fin. On peut noter que toutes ces configurations respectent l'encadrement désiré.

La figure 4.4 page 76 propose une illustration pour chacun des deux cas possibles lorsque $p_i \bmod \Delta = 0$, i.e. p_i est un multiple de Δ (dans la figure, le ratio $\frac{p_i}{\Delta}$ vaut

3). On peut noter que, dans le premier cas (début et fin non confondus avec une borne de période), il existe une seule configuration de début/fin, tandis que, dans le second cas (début et fin confondus avec une borne de période), il existe quatre configurations de début/fin, dont deux respectent l'encadrement désiré (la seconde et la troisième).

On peut remarquer qu'il existe plusieurs configurations de début/fin seulement lorsque la date de début ou la date de fin coïncide avec une borne entre deux périodes consécutives (cas limites). Deux conventions sont a priori envisageables : une telle borne appartient soit à la période précédente, soit à la période suivante. D'une part, imposer qu'une même convention (peu importe laquelle) soit appliquée pour toutes les bornes est en fait équivalent à interdire les configurations qui ne respectent pas l'encadrement désiré. D'autre part, choisir l'une ou l'autre des deux conventions revient à n'autoriser qu'une seule configuration de début/fin dans chaque cas. □

Grâce à cette observation, il sera possible de diminuer le nombre de variables sujettes à une contrainte d'intégrité explicite (variables permettant d'encoder les valeurs de ℓ_i^S et ℓ_i^C) dans les modèles mathématiques présentés dans la suite de ce chapitre.

- Si $p_i \bmod \Delta \neq 0$, alors une variable binaire $\pi_i \in \{0, 1\}$ peut être introduite pour caractériser la différence entre ℓ_i^S et ℓ_i^C.
  - $\pi_i = 0 \quad \Leftrightarrow \quad \ell_i^C - \ell_i^S = \lfloor \frac{p_i}{\Delta} \rfloor$
  - $\pi_i = 1 \quad \Leftrightarrow \quad \ell_i^C - \ell_i^S = \lceil \frac{p_i}{\Delta} \rceil$

  Autrement dit : $\ell_i^C - \ell_i^S = \lfloor \frac{p_i}{\Delta} \rfloor + \pi_i$

  Dans la figure 4.3, la valeur de la variable π_i est indiquée dans tous les cas.
- Si $p_i \bmod \Delta = 0$, alors il est possible d'exprimer directement l'un des indices en fonction de l'autre, puisque $\ell_i^C - \ell_i^S = \frac{p_i}{\Delta}$.

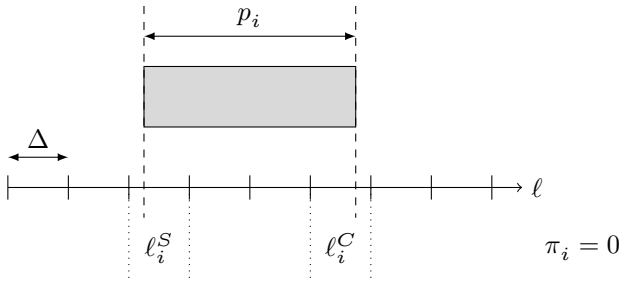
4.2 Modèles indexés par période

Dans cette section nous considérons des formulations originales à temps mixte. En effet, des variables continues sont utilisées pour représenter les dates de début des activités, tandis que des variables indexées par les périodes permettent de modéliser les contraintes de ressources agrégées. Dans le cas où le nombre de périodes n'est pas constant, ces modèles sont donc de taille pseudo-polynomiale. Nous proposons deux formulations, la deuxième exploitant la première propriété établie dans la section 4.1.1 et chacune pouvant être renforcée grâce à la deuxième propriété établie dans la section 4.1.2. Nous comparons sur le plan théorique la qualité des relaxations correspondantes, montrant que la deuxième formulation domine strictement la première.

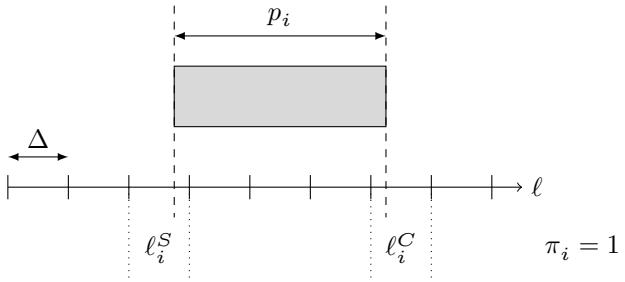
4.2.1 Première formulation

Description des variables

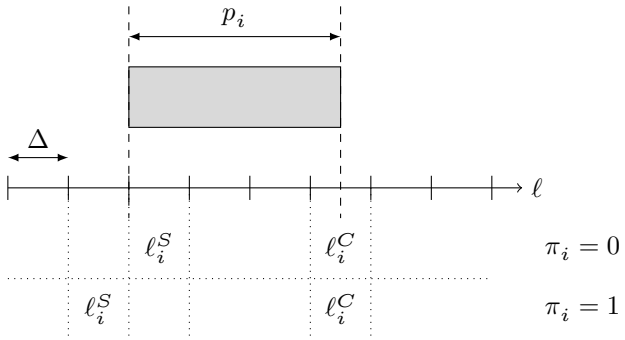
Les variables de décision utilisées dans ce modèle sont décrites ci-après dans le tableau 4.1. Une illustration des liens existant entre ces variables pour une activité donnée est disponible dans la figure 4.5.



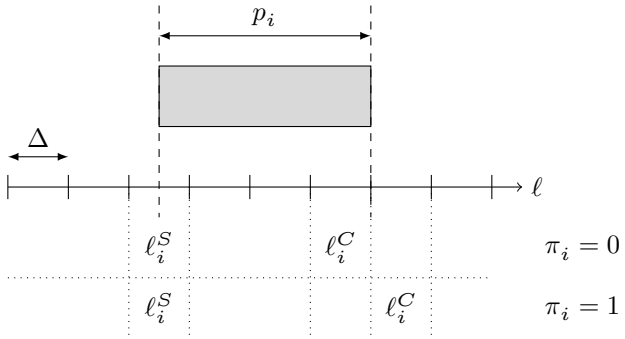
(a) Nombre de périodes intersectées minimal



(b) Nombre de périodes intersectées maximal

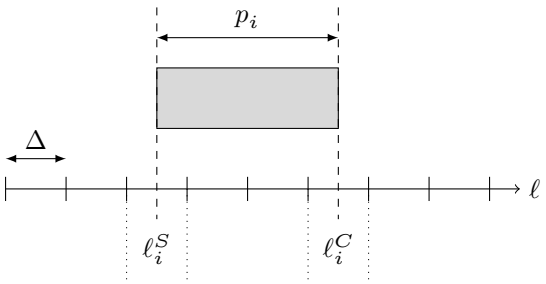


(c) Cas limite – Date de début confondue avec une borne de période

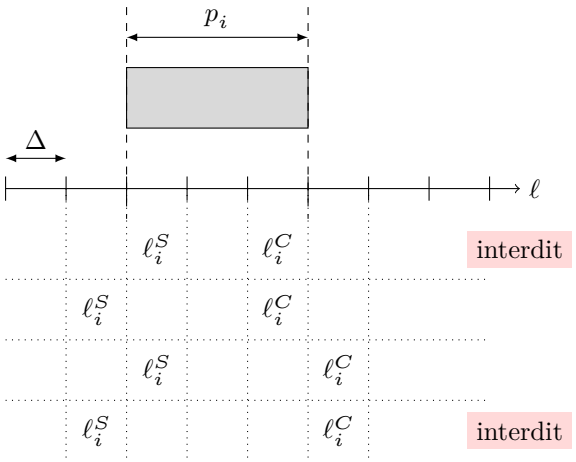


(d) Cas limite – Date de fin confondue avec une borne de période

Fig. 4.3 : Distance entre les indices de périodes de début et de fin lorsque la durée d'exécution d'une activité n'est pas un multiple de la durée des périodes



(a) Dates de début et de fin non confondues avec des bornes de période



(b) Cas limite – Dates de début et de fin confondues avec des bornes de période

Fig. 4.4 : Distance entre les indices de périodes de début et de fin lorsque la durée d'exécution d'une activité est un multiple de la durée des périodes

S_i	Date de début de l'exécution de l'activité $i \in \mathcal{A}$ S_0 (respectivement S_{n+1}) représente la date de début (respectivement de fin) d'exécution du projet.
$d_{i,\ell}$	Image de S_i par la fonction $D_{i,\ell}$
$zs_{i,\ell}$	Variables binaires avec un comportement *step* croissant ; encodage unaire de ℓ_i^S (indice de la période qui contient S_i)
$zf_{i,\ell}$	Variables binaires avec un comportement *step* croissant ; encodage unaire de ℓ_i^C (indice de la période qui contient S_i+p_i)

Tab. 4.1 : Variables de la première formulation indexée par les périodes

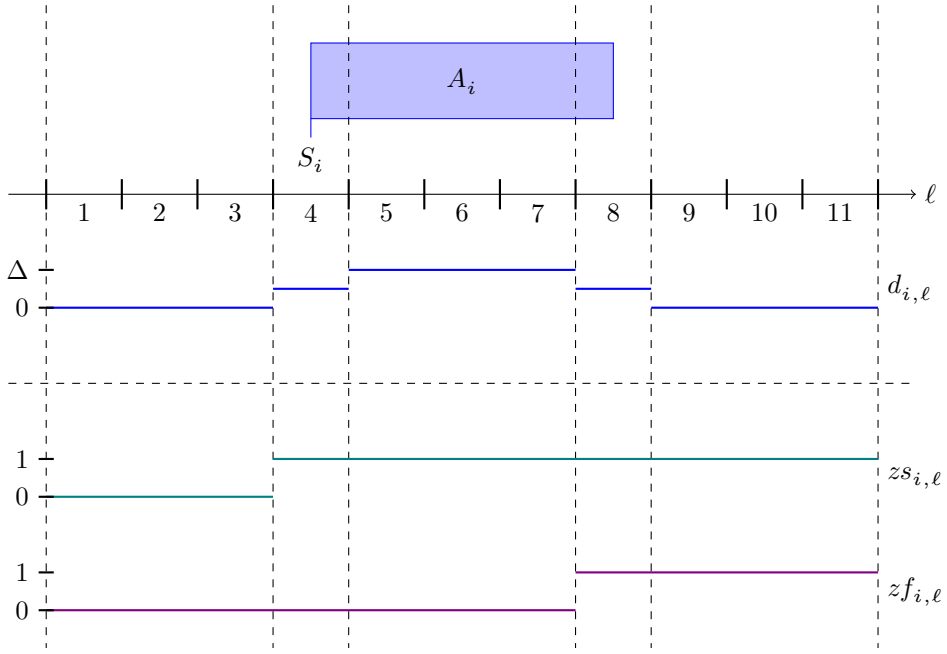


Fig. 4.5 : Représentation d'une fenêtre d'exécution à l'aide des variables de la première formulation indexée par période

Formulation initiale

$$\text{Minimiser} \quad S_{n+1} - S_0 \tag{4.2}$$

$$S_{i_2} - S_{i_1} \geq p_{i_1} \qquad \forall (i_1, i_2) \in E \tag{4.3}$$

$$\sum_{i \in \mathcal{A}} r_{i,k}\, d_{i,\ell} \leq b_k\, \Delta \qquad \forall k \in \mathcal{R}\ ,\ \forall \ell \in \mathcal{L} \tag{4.4}$$

$$S_i \geq \ell\Delta(1 - zs_{i,\ell}) \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{4.5}$$

$$S_i \leq L\Delta - (L - \ell)\Delta\, zs_{i,\ell} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{4.6}$$

$$S_i + p_i \geq \ell\Delta(1 - zf_{i,\ell}) \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{4.7}$$

$$S_i + p_i \leq L\Delta - (L-\ell)\Delta\, zf_{i,\ell} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.8)$$

$$d_{i,\ell} \geq 0 \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.9)$$

$$d_{i,\ell} \leq \Delta\,(zs_{i,\ell} - zf_{i,\ell-1}) \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.10)$$

$$d_{i,\ell} \geq \Delta\,(zs_{i,\ell-1} - zf_{i,\ell}) \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.11)$$

$$\begin{aligned} d_{i,\ell} \geq\ & \ell\Delta - S_i \\ & - \Delta\, zf_{i,\ell} \\ & - \ell\Delta\, zs_{i,\ell-1} \end{aligned} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.12)$$

$$\begin{aligned} d_{i,\ell} \geq\ & S_i + p_i - (\ell-1)\Delta \\ & - \Delta(1 - zs_{i,\ell-1}) \\ & - (L-\ell+1)\Delta(1 - zf_{i,\ell}) \end{aligned} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.13)$$

$$\sum_{\ell \in \mathcal{L}} d_{i,\ell} = p_i \qquad \forall i \in \mathcal{A} \qquad (4.14)$$

$$zs_{i,\ell-1} \leq zs_{i,\ell} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.15)$$

$$zf_{i,\ell-1} \leq zf_{i,\ell} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.16)$$

$$zs_{i,\ell} \in \{0,1\} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.17)$$

$$zf_{i,\ell} \in \{0,1\} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.18)$$

L'objectif est de minimiser la durée du projet (4.2), sous contraintes d'antériorité (4.3) et sous contraintes de ressources agrégées par période (4.4). Les contraintes 4.5 et 4.6 permettent de lier les variables S_i aux variables $zs_{i,\ell}$, tandis que les contraintes 4.7 et 4.8 permettent de lier la date $S_i + p_i$ aux variables $zf_{i,\ell}$.

Les contraintes suivantes permettent de calculer les valeurs des variables $d_{i,\ell}$.

- Les contraintes 4.9 fixent le signe des variables $d_{i,\ell}$.
- Les contraintes 4.10 forcent $d_{i,\ell}$ à prendre la valeur 0 lorsque la période ℓ est située soit avant soit après la fenêtre d'exécution de l'activité i.
- Les contraintes 4.11 forcent $d_{i,\ell}$ à prendre la valeur Δ lorsque la période ℓ est incluse en intégralité dans la fenêtre d'exécution de l'activité i.
- Les contraintes 4.12 permettent de calculer la valeur de $d_{i,\ell}$ lorsque $\ell = \ell_i^S$ (période qui contient S_i).
- Les contraintes 4.13 permettent de calculer la valeur de $d_{i,\ell}$ lorsque $\ell = \ell_i^C$ (période qui contient $S_i + p_i$).
- Les contraintes 4.14 permettent de prendre en compte la durée de l'activité i.

Enfin, les contraintes 4.15 et 4.16 assurent un comportement *step* croissant des variables binaires (contraintes 4.17 et 4.18).

Renforcement de la formulation

Puisque la subdivision temporelle considérée est uniforme (périodes de même durée), il est possible de limiter l'espace des solutions de sorte que le projet commence dans la première période.

$$0 \leq S_0 \leq \Delta \qquad (4.19)$$

De plus, puisque la préemption n'est pas autorisée, le nombre de périodes intersectées par une fenêtre d'exécution est borné (cf. deuxième remarque préliminaire). Ainsi, il est possible de lier les variables $zs_{i,\ell}$ et $zf_{i,\ell}$ via une variable binaire π_i (une seule variable binaire par activité), de sorte que :

$$\begin{aligned} \pi_i = 0 \quad &\Leftrightarrow \quad zs_{i,\ell} = zf_{i,\ell+\lfloor \frac{p_i}{\Delta} \rfloor} \\ \pi_i = 1 \quad &\Leftrightarrow \quad zs_{i,\ell} = zf_{i,\ell+\lceil \frac{p_i}{\Delta} \rceil} \end{aligned}$$

Dans ce cas, il est possible de relâcher la contrainte d'intégrité explicite sur l'une des deux familles de variables binaires *step*, par exemple les variables $zf_{i,\ell}$. Cela permet de diminuer considérablement le nombre de variables entières (environ deux fois moins lorsque le nombre de périodes est élevé).

$$zf_{i,\ell} \in [0,1] \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.20)$$

$$\pi_i \in \{0,1\} \qquad \forall i \in \mathcal{A} \qquad (4.21)$$

$$zs_{i,\ell} \geq zf_{i,\ell+\lfloor \frac{p_i}{\Delta} \rfloor} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.22)$$

$$zs_{i,\ell} \leq zf_{i,\ell+\lceil \frac{p_i}{\Delta} \rceil} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.23)$$

$$zs_{i,\ell} \leq zf_{i,\ell+\lfloor \frac{p_i}{\Delta} \rfloor} + \pi_i \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.24)$$

$$zs_{i,\ell} \geq zf_{i,\ell+\lceil \frac{p_i}{\Delta} \rceil} + \pi_i - 1 \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.25)$$

Remarque. Si une activité $i \in \mathcal{A}$ est telle que $p_i \bmod \Delta = 0$, il est en fait inutile d'introduire la variable π_i pour cette activité, car il existe alors une bijection directe entre les variables $zs_{i,\ell}$ et $zf_{i,\ell}$, comme cela a été démontré dans l'analyse préliminaire.

$$zs_{i,\ell} = zf_{i,\ell+\frac{p_i}{\Delta}} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.26)$$

4.2.2 Seconde formulation

La seconde formulation est basée sur l'analyse structurelle effectuée dans la section 4.1.1 et sur la décomposition de la fonction d'intersection de la fenêtre temporelle de l'activité avec la période en fonctions monotones. Nous verrons par la suite que cette décomposition permet d'obtenir une formulation strictement dominante en termes de qualité de la relaxation linéaire.

Description des variables

Les variables de décision utilisées dans ce modèle sont décrites ci-après dans le tableau 4.2. Une illustration des liens existant entre ces variables pour une activité donnée est disponible dans la figure 4.6.

S_i	Date de début de l'exécution de l'activité $i \in \mathcal{A}$ S_0 (respectivement S_{n+1}) représente la date de début (respectivement de fin) d'exécution du projet.
$d_{i,\ell}$	Image de S_i par la fonction $D_{i,\ell}$
$\lambda_{i,\ell}$	Image de S_i par la fonction $\Lambda_{i,\ell}$
$\mu_{i,\ell}$	Image de S_i par la fonction $\mathrm{M}_{i,\ell}$
$z^{\lambda}_{i,\ell}$	Variables binaires permettant de forcer un comportement *step* décroissant des variables $\lambda_{i,\ell}$
$z^{\mu}_{i,\ell}$	Variables binaires permettant de forcer un comportement *step* croissant des variables $\mu_{i,\ell}$

Tab. 4.2 : Variables de la seconde formulation indexée par les périodes

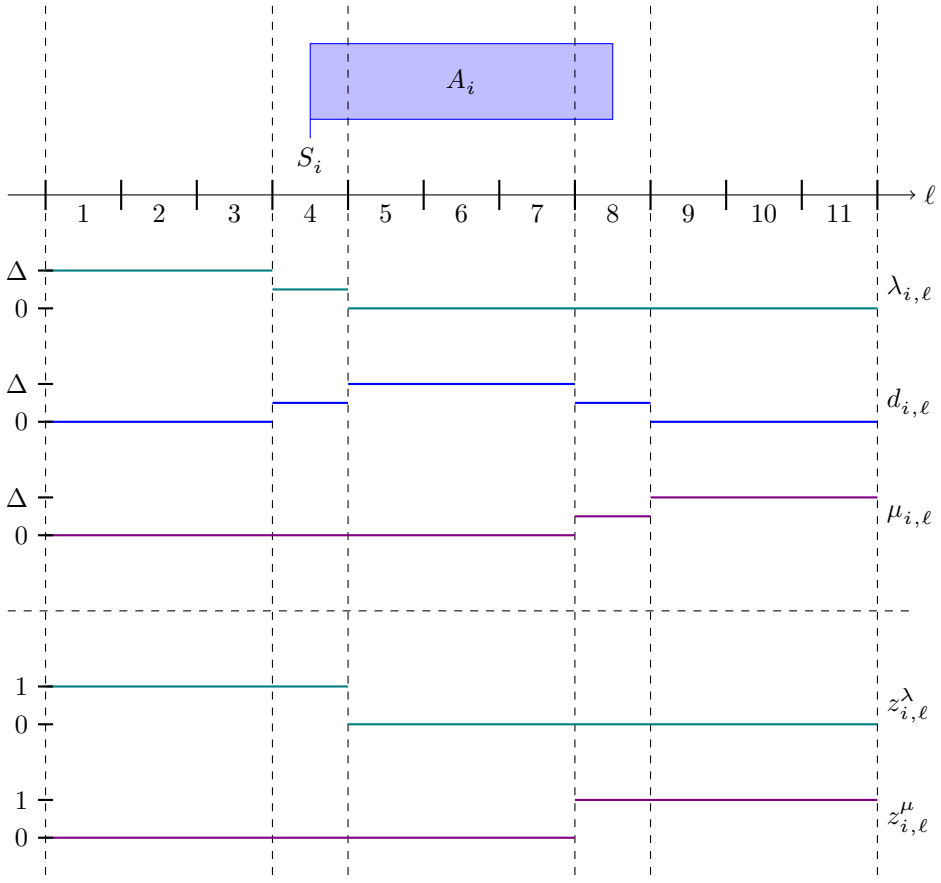


Fig. 4.6 : Représentation d'une fenêtre d'exécution à l'aide des variables de la seconde formulation indexée par période

Formulation initiale

$$\text{Minimiser} \quad S_{n+1} - S_0 \tag{4.27}$$

$$S_{i_2} - S_{i_1} \geq p_{i_1} \qquad \forall (i_1, i_2) \in E \qquad (4.28)$$

$$\sum_{i \in \mathcal{A}} r_{i,k} d_{i,\ell} \leq b_k \Delta \qquad \forall k \in \mathcal{R}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.29)$$

$$\lambda_{i,\ell} + d_{i,\ell} + \mu_{i,\ell} = \Delta \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.30)$$

$$S_i = \sum_{\ell \in \mathcal{L}} \lambda_{i,\ell} \qquad \forall i \in \mathcal{A} \qquad (4.31)$$

$$\sum_{\ell \in \mathcal{L}} d_{i,\ell} = p_i \qquad \forall i \in \mathcal{A} \qquad (4.32)$$

$$\lambda_{i,\ell} \leq \Delta\, z^{\lambda}_{i,\ell} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.33)$$

$$\lambda_{i,\ell} \geq \Delta\, z^{\lambda}_{i,\ell+1} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.34)$$

$$\mu_{i,\ell} \leq \Delta\, z^{\mu}_{i,\ell} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.35)$$

$$\mu_{i,\ell} \geq \Delta\, z^{\mu}_{i,\ell-1} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.36)$$

$$z^{\lambda}_{i,\ell} \in \{0,1\} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.37)$$

$$z^{\mu}_{i,\ell} \in \{0,1\} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.38)$$

$$d_{i,\ell} \geq 0 \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \qquad (4.39)$$

L'objectif est de minimiser la durée du projet (4.27), sous contraintes d'antériorité (4.28) et sous contraintes de ressources agrégées par période (4.29). Les contraintes 4.30 traduisent la relation de partition au sein d'une période. Les contraintes 4.31 permettent de calculer S_i à partir des $\lambda_{i,\ell}$. Les contraintes 4.32 permettent de prendre en compte les durées des activités.

Les contraintes 4.33 et 4.34 permettent de définir un comportement *step* croisé entre les variables $\lambda_{i,\ell}$ et $z^{\lambda}_{i,\ell}$, de telle sorte qu'une seule variable $\lambda_{i,\ell}$ peut varier entre 0 et Δ, alors que les autres prennent soit la valeur 0, soit la valeur Δ. Les contraintes 4.35 et 4.36 fonctionnent de manière analogue avec les variables $\mu_{i,\ell}$ et $z^{\mu}_{i,\ell}$ (cf figure 4.6).

Enfin, les variables $z^{\lambda}_{i,\ell}$ et $z^{\mu}_{i,\ell}$ sont binaires (contraintes 4.37 et 4.38) tandis que les variables $d_{i,\ell}$ sont positives (contraintes 4.39).

Renforcement de la formulation

Grâce à l'introduction des variables $\lambda_{i,\ell}$ et $\mu_{i,\ell}$, il devient possible de désagréger les contraintes d'antériorité, comme cela est illustré dans la figure 4.7.

$$\mu_{i_1,\ell} + \lambda_{i_2,\ell} \geq \Delta \qquad \forall (i_1, i_2) \in E\ ,\ \forall \ell \in \mathcal{L} \qquad (4.40)$$

Comme précédemment, il est possible de forcer l'exécution du projet à avoir lieu dans la première période.

$$0 \leq S_0 \leq \Delta \qquad (4.41)$$

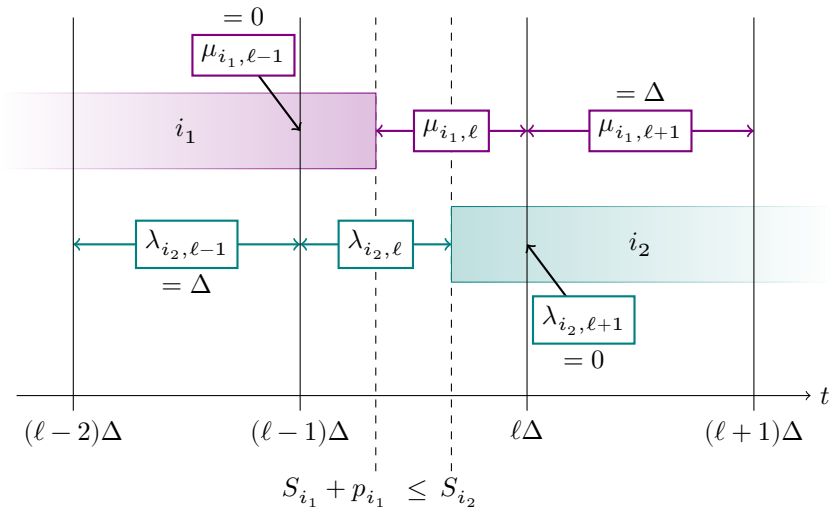


Fig. 4.7 : Désagrégation des contraintes d'antériorité

Comme précédemment, il est possible d'introduire une variable binaire π_i par activité, afin de renforcer le lien entre les variables binaires *step*, tout en réduisant le nombre de variables soumises à une contrainte d'intégrité explicite.

$$\begin{aligned} \pi_i = 0 \quad &\Leftrightarrow \quad z^{\lambda}_{i,\ell} + z^{\mu}_{i,\ell+\lfloor \frac{p_i}{\Delta} \rfloor - 1} = 1 \\ \pi_i = 1 \quad &\Leftrightarrow \quad z^{\lambda}_{i,\ell} + z^{\mu}_{i,\ell+\lceil \frac{p_i}{\Delta} \rceil - 1} = 1 \end{aligned}$$

$$z^{\mu}_{i,\ell} \in [0,1] \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{4.42}$$

$$\pi_i \in \{0,1\} \qquad \forall i \in \mathcal{A} \tag{4.43}$$

$$z^{\lambda}_{i,\ell} + z^{\mu}_{i,\ell+\lfloor \frac{p_i}{\Delta} \rfloor - 1} \leq 1 \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{4.44}$$

$$z^{\lambda}_{i,\ell} + z^{\mu}_{i,\ell+\lceil \frac{p_i}{\Delta} \rceil - 1} \geq 1 \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{4.45}$$

$$z^{\lambda}_{i,\ell} + z^{\mu}_{i,\ell+\lfloor \frac{p_i}{\Delta} \rfloor - 1} \geq 1 - \pi_i \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{4.46}$$

$$z^{\lambda}_{i,\ell} + z^{\mu}_{i,\ell+\lceil \frac{p_i}{\Delta} \rceil - 1} \leq 2 - \pi_i \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{4.47}$$

Remarque. Si une activité $i \in \mathcal{A}$ est telle que $p_i \bmod \Delta = 0$, alors il est inutile d'introduire la variable π_i pour cette activité, car il est alors possible d'exprimer directement certaines variables en fonction d'autres.

$$z^{\lambda}_{i,\ell} + z^{\mu}_{i,\ell+\frac{p_i}{\Delta}-1} = 1 \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{4.48}$$

$$\lambda_{i,\ell} + \mu_{i,\ell+\frac{p_i}{\Delta}} = \Delta \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{4.49}$$

4.2.3 Comparaison des deux formulations

Proposition 4.3. *La relaxation linéaire de la seconde formulation est meilleure que celle de la première formulation.*

Démonstration. Remarquons tout d'abord qu'il existe des bijections linéaires entre les variables binaires du premier modèle ($zs_{i,\ell}$ et $zf_{i,\ell}$) et celles du second modèle ($z^{\lambda}_{i,\ell}$ et $z^{\mu}_{i,\ell}$).

$$zs_{i,\ell} = 1 - z^{\lambda}_{i,\ell+1}$$

$$zf_{i,\ell} = z^{\mu}_{i,\ell}$$

Concernant les variables continues, les unes apparaissent dans les deux modèles (S_i et $d_{i,\ell}$) avec la même signification, tandis que les autres ($\lambda_{i,\ell}$ et $\mu_{i,\ell}$) apparaissent seulement dans le second modèle.

Rappelons qu'il est possible d'exprimer la date de début d'exécution de l'activité i en fonction des variables $\lambda_{i,\ell}$ uniquement (contraintes 4.31).

$$S_i = \sum_{\ell=1}^{L} \lambda_{i,\ell}$$

De manière similaire, il est possible d'exprimer la date de début d'exécution de l'activité i en fonction des variables $\mu_{i,\ell}$ uniquement (en utilisant les contraintes (4.30) à (4.32)).

$$\begin{aligned} S_i + p_i &= \left(\sum_{\ell=1}^{L} \lambda_{i,\ell}\right) + \left(\sum_{\ell=1}^{L} d_{i,\ell}\right) \\ &= \sum_{\ell=1}^{L} \left(\lambda_{i,\ell} + d_{i,\ell}\right) \\ &= \sum_{\ell=1}^{L} \left(\Delta - \mu_{i,\ell}\right) \\ &= L\Delta - \sum_{\ell=1}^{L} \mu_{i,\ell} \end{aligned}$$

Il reste à montrer que l'enveloppe convexe définie par les contraintes du second modèle est incluse dans l'enveloppe convexe définie par les contraintes du premier modèle.

Concernant les contraintes d'antériorité, celles-ci s'écrivent soit de manière agrégée, avec la même expression dans les deux modèles (contraintes 4.3 et 4.28), soit de manière désagrégée, uniquement dans le second modèle (contraintes 4.40). On remarque que, pour une relation d'antériorité donnée, la contrainte sous forme agrégée peut être obtenue en sommant les contraintes sous forme désagrégée sur l'ensemble des périodes (cf. expressions des dates de début et de fin ci-avant).

De plus, les contraintes restantes du premier modèle, récrites en substituant les variables binaires du premier modèle par celles du second, peuvent être obtenues à partir des contraintes du second modèle. Les démonstrations pour les contraintes (4.5) à (4.8) (lien entre S_i, $zs_{i,\ell}$ et $zf_{i,\ell}$) sont regroupées dans le tableau 4.3 page 84, tandis que les démonstratations pour les contraintes (4.10) à (4.13) (calcul de $d_{i,\ell}$) sont regroupées dans le tableau 4.4 page 85. Les contraintes (4.9) et (4.14) du premier modèle sont aussi présentes dans le second modèle (contraintes 4.39 et 4.32).

Enfin, les contraintes (4.22) à (4.26) permettant de renforcer le premier modèle en liant les variables $zs_{i,\ell}$ et $zf_{i,\ell}$ via une variable binaire supplémentaire π_i sont équivalentes aux contraintes (4.44) à (4.48) permettant de renforcer le second modèle en liant les variables $z^{\lambda}_{i,\ell}$ et $z^{\mu}_{i,\ell}$ via la même variable binaire π_i : pour passer des unes aux autres, il suffit de changer de variables en utilisant les bijections linéaires entre variables *step*. □

Proposition 4.4. *La relaxation linéaire de la seconde formulation est* strictement *meilleure que celle de la première formulation.*

$$S_i \leq (\ell-1)\Delta + (L-\ell+1)\Delta z^{\lambda}_{i,\ell} \tag{4.5'}$$

$$\begin{aligned}
&S_i - (\ell-1)\Delta - (L-\ell+1)\Delta z^{\lambda}_{i,\ell} \\
&= -(\ell-1)\Delta + \left(\textstyle\sum_{\ell'=1}^{L} \lambda_{i,\ell'}\right) - \left(\textstyle\sum_{\ell'=\ell}^{L} \Delta\right) z^{\lambda}_{i,\ell} \\
&\leq -\textstyle\sum_{\ell'=\ell}^{L} \left(\Delta z^{\lambda}_{i,\ell'} - \lambda_{i,\ell'}\right) - \sum_{\ell'=1}^{\ell-1} \left(\Delta - \lambda_{i,\ell'}\right) \\
&\leq 0
\end{aligned}$$

$$S_i \geq (\ell-1)\Delta - (\ell-1)\Delta \left(1 - z^{\lambda}_{i,\ell}\right) \tag{4.6'}$$

$$\begin{aligned}
&S_i - (\ell-1)\Delta + (\ell-1)\Delta \left(1 - z^{\lambda}_{i,\ell}\right) \\
&= -(\ell-1)\Delta + \left(\textstyle\sum_{\ell'=1}^{L} \lambda_{i,\ell'}\right) + (\ell-1)\Delta - \left(\textstyle\sum_{\ell'=1}^{\ell-1} \Delta\right) z^{\lambda}_{i,\ell} \\
&\geq \textstyle\sum_{\ell'=1}^{L} \lambda_{i,\ell'} - \sum_{\ell'=1}^{\ell-1} \Delta z^{\lambda}_{i,\ell'+1} \\
&\geq \textstyle\sum_{\ell'=1}^{L} \lambda_{i,\ell'} - \sum_{\ell'=1}^{\ell-1} \lambda_{i,\ell'} \\
&= \textstyle\sum_{\ell'=\ell}^{L} \lambda_{i,\ell'} \\
&\geq 0
\end{aligned}$$

$$S_i + p_i \leq \ell\Delta + (L-\ell)\Delta \left(1 - z^{\mu}_{i,\ell}\right) \tag{4.7'}$$

$$\begin{aligned}
&S_i + p_i - \ell\Delta - (L-\ell)\Delta \left(1 - z^{\mu}_{i,\ell}\right) \\
&= -\ell\Delta + \left(L\Delta - \textstyle\sum_{\ell'=1}^{L} \mu_{i,\ell'}\right) - (L-\ell)\Delta + \left(\textstyle\sum_{\ell'=\ell+1}^{L} \Delta\right) z^{\mu}_{i,\ell} \\
&\leq -\textstyle\sum_{\ell'=1}^{L} \mu_{i,\ell'} + \sum_{\ell'=\ell+1}^{L} \Delta z^{\mu}_{i,\ell'-1} \\
&\leq -\textstyle\sum_{\ell'=1}^{L} \mu_{i,\ell'} + \sum_{\ell'=\ell+1}^{L} \mu_{i,\ell'} \\
&= -\textstyle\sum_{\ell'=1}^{\ell} \mu_{i,\ell'} \\
&\leq 0
\end{aligned}$$

$$S_i + p_i \geq \ell\Delta - \ell\Delta z^{\mu}_{i,\ell} \tag{4.8'}$$

$$\begin{aligned}
&S_i + p_i - \ell\Delta + \ell\Delta z^{\mu}_{i,\ell} \\
&= -\ell\Delta + \left(L\Delta - \textstyle\sum_{\ell'=1}^{L} \mu_{i,\ell'}\right) + \left(\textstyle\sum_{\ell'=1}^{\ell} \Delta\right) z^{\mu}_{i,\ell} \\
&\geq \textstyle\sum_{\ell'=1}^{\ell} \left(\Delta z^{\mu}_{i,\ell'} - \mu_{i,\ell'}\right) + \sum_{\ell'=\ell+1}^{L} \left(\Delta - \mu_{i,\ell'}\right) \\
&\geq 0
\end{aligned}$$

Tab. 4.3 : Comparaison des modèles indexés par les périodes (1/2)

$$d_{i,\ell} \geq \Delta \left(1 - z^{\lambda}_{i,\ell} - z^{\mu}_{i,\ell}\right) \tag{4.10'}$$

$$\begin{aligned}
& d_{i,\ell} - \Delta \left(1 - z^{\lambda}_{i,\ell} - z^{\mu}_{i,\ell}\right) \\
&= \Delta - \lambda_{i,\ell} - \mu_{i,\ell} - \Delta + \Delta z^{\lambda}_{i,\ell} + \Delta z^{\mu}_{i,\ell} \\
&= \left(\Delta z^{\lambda}_{i,\ell} - \lambda_{i,\ell}\right) + \left(\Delta z^{\mu}_{i,\ell} - \mu_{i,\ell}\right) \\
&\geq 0
\end{aligned}$$

$$d_{i,\ell} \leq \Delta \left(1 - z^{\lambda}_{i,\ell+1} - z^{\mu}_{i,\ell-1}\right) \tag{4.11'}$$

$$\begin{aligned}
& d_{i,\ell} - \Delta \left(1 - z^{\lambda}_{i,\ell+1} - z^{\mu}_{i,\ell-1}\right) \\
&= \Delta - \lambda_{i,\ell} - \mu_{i,\ell} - \Delta + \Delta z^{\lambda}_{i,\ell+1} + \Delta z^{\mu}_{i,\ell-1} \\
&= \left(\Delta z^{\lambda}_{i,\ell+1} - \lambda_{i,\ell}\right) + \left(\Delta z^{\mu}_{i,\ell-1} - \mu_{i,\ell}\right) \\
&\leq 0
\end{aligned}$$

$$d_{i,\ell} \geq \ell\Delta - S_i - \Delta z^{\mu}_{i,\ell} - \ell\Delta \left(1 - z^{\lambda}_{i,\ell}\right) \tag{4.12'}$$

$$\begin{aligned}
& d_{i,\ell} - \ell\Delta + S_i + \Delta z^{\mu}_{i,\ell} + \ell\Delta \left(1 - z^{\lambda}_{i,\ell}\right) \\
&= d_{i,\ell} - \ell\Delta + \left(\textstyle\sum_{\ell'=1}^{L} \lambda_{i,\ell'}\right) + \Delta z^{\mu}_{i,\ell} + \ell\Delta - \ell\Delta z^{\lambda}_{i,\ell} \\
&= \textstyle\sum_{\ell'=1}^{L} \lambda_{i,\ell'} + d_{i,\ell} + \Delta \left(z^{\mu}_{i,\ell} - z^{\lambda}_{i,\ell}\right) - \left(\sum_{\ell'=1}^{\ell-1} \Delta\right) z^{\lambda}_{i,\ell} \\
&\geq \textstyle\sum_{\ell'=1}^{L} \lambda_{i,\ell'} + (\Delta - \lambda_{i,\ell} - \mu_{i,\ell}) + \Delta \left(z^{\mu}_{i,\ell} - z^{\lambda}_{i,\ell}\right) - \left(\sum_{\ell'=1}^{\ell-1} \Delta\ z^{\lambda}_{i,\ell'+1}\right) \\
&\geq \left(\textstyle\sum_{\ell'=1}^{L} \lambda_{i,\ell'} - \lambda_{i,\ell} - \sum_{\ell'=1}^{\ell-1} \lambda_{i,\ell'}\right) + \left(\Delta z^{\mu}_{i,\ell} - \mu_{i,\ell}\right) + \Delta \left(1 - z^{\lambda}_{i,\ell}\right) \\
&\geq \textstyle\sum_{\ell'=\ell+1}^{L} \lambda_{i,\ell'} \\
&\geq 0
\end{aligned}$$

$$d_{i,\ell} \geq S_i + p_i - (\ell - 1)\Delta - \Delta z^{\lambda}_{i,\ell} - (L - \ell + 1)\Delta \left(1 - z^{\mu}_{i,\ell}\right) \tag{4.13'}$$

$$\begin{aligned}
& d_{i,\ell} - S_i - p_i + (\ell - 1)\Delta + \Delta z^{\lambda}_{i,\ell} + (L - \ell + 1)\Delta \left(1 - z^{\mu}_{i,\ell}\right) \\
&= d_{i,\ell} + (\ell - 1)\Delta - \left(L\Delta - \textstyle\sum_{\ell'=1}^{L} \mu_{i,\ell'}\right) + \Delta z^{\lambda}_{i,\ell} + (L - \ell + 1)\Delta - (L - \ell + 1)\Delta z^{\mu}_{i,\ell} \\
&= \textstyle\sum_{\ell'=1}^{L} \mu_{i,\ell'} + d_{i,\ell} + \Delta \left(z^{\lambda}_{i,\ell} - z^{\mu}_{i,\ell}\right) - \left(\sum_{\ell'=\ell+1}^{L} \Delta\right) z^{\mu}_{i,\ell} \\
&\geq \textstyle\sum_{\ell'=1}^{L} \mu_{i,\ell'} + (\Delta - \lambda_{i,\ell} - \mu_{i,\ell}) + \Delta \left(z^{\lambda}_{i,\ell} - z^{\mu}_{i,\ell}\right) - \left(\sum_{\ell'=\ell+1}^{L} \Delta\ z^{\mu}_{i,\ell'-1}\right) \\
&\geq \left(\textstyle\sum_{\ell'=1}^{L} \mu_{i,\ell'} - \mu_{i,\ell} - \sum_{\ell'=\ell+1}^{L} \mu_{i,\ell'}\right) + \left(\Delta z^{\lambda}_{i,\ell} - \lambda_{i,\ell}\right) + \Delta \left(1 - z^{\mu}_{i,\ell}\right) \\
&\geq \textstyle\sum_{\ell'=1}^{\ell-1} \mu_{i,\ell'} \\
&\geq 0
\end{aligned}$$

Tab. 4.4 : Comparaison des modèles indexés par les périodes (2/2)

Démonstration. Pour certaines instances du PARCPSP, la valeur de la relaxation de la seconde formulation est en effet strictement supérieure à celle de la première. C'est par exemple le cas pour l'instance construite avec le projet 6 de la classe 29 parmi les instances à 30 activités et 4 ressources (J30) de la PSPLIB (Kolisch et Sprecher 1996) et des périodes de durée $\Delta = 1$: la borne retournée par le premier modèle est inférieure à 44, tandis que la borne retournée par le second modèle est supérieure à 58 ; l'écart relatif est donc supérieur à 34% (valeur de l'écart relatif maximal dans la ligne $\Delta = 1$ du tableau 5.4b page 107). □

4.3 Modèle compact

L'inconvénient des formulations indexées par les périodes est la possible explosion de la taille des formulations lorsque l'horizon de planification est grand ou lorsque la durée des périodes Δ est petite. La question de l'existence d'une formulation compacte (avec un nombre polynomial de variables et de contraintes, ne dépendant donc pas du nombre de périodes) a un intérêt théorique mais aussi, de ce fait, possiblement pratique. Dans cette section, nous proposons une telle formulation pour le PARCPSP. Pour cela, nous nous basons sur des variables entières qui permettent d'identifier les bornes de périodes situées autour de la fenêtre d'exécution d'une activité, ainsi que des variables continues représentant la distance des dates de début et de fin par rapport à ces bornes. Nous donnons dans un premier temps une formulation non linéaire en nombres entiers, puis nous montrons comment la linéariser afin d'obtenir la formulation compacte.

Rappel – Vérification en temps polynomial de la faisabilité d'une solution

Dans le chapitre 2, nous avons montré que le problème de décision associé au PARCPSP est **NP**-difficile. En effet, l'algorithme 2.1 vérifie en temps polynomial si une solution est réalisable ou non. En particulier, la complexité polynomiale de l'algorithme découle du fait qu'il est nécessaire de tester les contraintes de ressources agrégées seulement dans certaines périodes bien précises : pour chaque activité, il faut toujours tester la période dans laquelle cette activité commence, et parfois la période suivante.

En utilisant cette propriété, il devient possible d'écrire une formulation exacte pour le PARCPSP sous la forme d'un programme linéaire en nombres entiers compact, i.e. de taille non plus pseudo-polynomiale mais polynomiale.

Description des variables

Les variables de décision utilisées dans ce modèle sont décrites ci-après dans le tableau 4.5. Une illustration des liens existant entre ces variables pour une activité donnée est disponible dans la figure 4.8.

S_i	Date de début d'exécution de l'activité $i \in \mathcal{A}$
ℓ_i^S	Variable entière ($i \in \mathcal{A}$) représentant l'indice de la période qui contient S_i moins 1 (borne de gauche)
ℓ_i^C	Variable entière ($i \in \mathcal{A}$) représentant l'indice de la période qui contient $S_i + p_i$ (borne de droite)
τ_i^S	Variable continue ($i \in \mathcal{A}$) égale à $S_i - \ell_i^S \Delta$ (distance à la borne de gauche de la période qui contient S_i)
τ_i^C	Variable continue ($i \in \mathcal{A}$) égale à $\ell_i^C \Delta - C_i$ (distance à la borne de gauche de la période qui contient $S_i + p_i$)
$d_{i',i,\alpha}$	Variable continue égale à $D_{i',\ell}(S_{i'})$ ($i' \in \mathcal{A}$) où $\ell = \ell_i^S + \alpha$ ($i \in \mathcal{A}$, $\alpha \in \{0,1\}$)
$\lambda_{i',i,\alpha}$	Variable continue égale à $\Lambda_{i',\ell}(S_{i'})$ ($i' \in \mathcal{A}$) où $\ell = \ell_i^S + \alpha$ ($i \in \mathcal{A}$, $\alpha \in \{0,1\}$)
$\mu_{i',i,\alpha}$	Variable continue égale à $\mathrm{M}_{i',\ell}(S_{i'})$ ($i' \in \mathcal{A}$) où $\ell = \ell_i^S + \alpha$ ($i \in \mathcal{A}$, $\alpha \in \{0,1\}$)
$\mathbf{1}_X$	Notation générique ; fonction binaire prenant la valeur 1 si l'expression booléenne X est vraie, 0 sinon

Tab. 4.5 : Variables de la formulation compacte

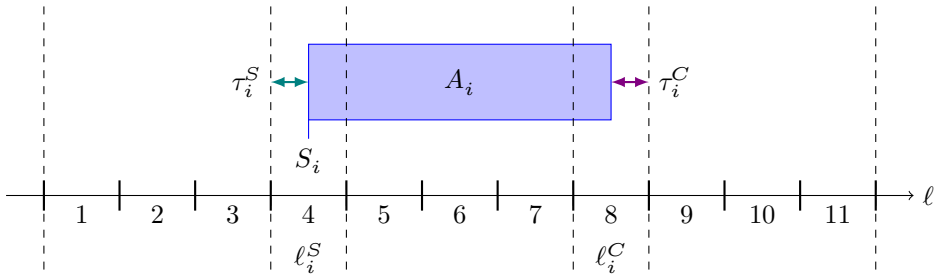


Fig. 4.8 : Représentation d'une fenêtre d'exécution à l'aide des variables de la formulation compacte

Formulation initiale non linéaire

La formulation proposée ci-après permet de comprendre comment fonctionne le modèle compact. En l'état, elle n'est pas linéaire ; nous montrerons par la suite qu'il est néanmoins possible de la linéariser.

$$\text{Minimiser} \quad S_{n+1} - S_0 \tag{4.50}$$

$$S_{i_2} - S_{i_1} \geq p_{i_1} \qquad \forall (i_1, i_2) \in E \tag{4.51}$$

$$\sum_{i' \in \mathcal{A}} r_{i',k}\, d_{i',i,\alpha} \leq b_k\, \Delta \qquad \forall k \in \mathcal{R} \quad \forall i \in \mathcal{A} \quad \forall \alpha \in \{0,1\} \tag{4.52}$$

$$S_i = (\ell_i^S - 1)\Delta + \tau_i^S \qquad \forall i \in \mathcal{A} \tag{4.53}$$

$$S_i + p_i = \ell_i^C \Delta - \tau_i^C \qquad \forall i \in \mathcal{A} \tag{4.54}$$

$$\lambda_{i',i,\alpha} + d_{i',i,\alpha} + \mu_{i',i,\alpha} = \Delta \qquad \forall i' \in \mathcal{A} \quad \forall i \in \mathcal{A} \quad \forall \alpha \in \{0,1\} \tag{4.55}$$

$$d_{i',i,\alpha} \geq 0 \qquad \forall i' \in \mathcal{A} \quad \forall i \in \mathcal{A} \quad \forall \alpha \in \{0,1\} \tag{4.56}$$

$$\begin{aligned} \lambda_{i',i,\alpha} = \quad & 0 \times \mathbf{1}_{\ell_{i'}^S < \ell_i^S + \alpha} \\ & + \tau_{i'}^S \times \mathbf{1}_{\ell_{i'}^S = \ell_i^S + \alpha} \\ & + \Delta \times \mathbf{1}_{\ell_{i'}^S > \ell_i^S + \alpha} \end{aligned} \qquad \forall i' \in \mathcal{A} \quad \forall i \in \mathcal{A} \quad \forall \alpha \in \{0,1\} \tag{4.57}$$

$$\begin{aligned} \mu_{i',i,\alpha} = \quad & \Delta \times \mathbf{1}_{\ell_{i'}^C < \ell_i^S + \alpha} \\ & + \tau_{i'}^C \times \mathbf{1}_{\ell_{i'}^C = \ell_i^S + \alpha} \\ & + 0 \times \mathbf{1}_{\ell_{i'}^C > \ell_i^S + \alpha} \end{aligned} \qquad \forall i' \in \mathcal{A} \quad \forall i \in \mathcal{A} \quad \forall \alpha \in \{0,1\} \tag{4.58}$$

$$\ell_i^S \in \{1, \dots, L\} \qquad \forall i \in \mathcal{A} \tag{4.59}$$

$$\ell_i^C \in \{1, \dots, L\} \qquad \forall i \in \mathcal{A} \tag{4.60}$$

$$0 \leq \tau_i^S \leq \Delta \qquad \forall i \in \mathcal{A} \tag{4.61}$$

$$0 \leq \tau_i^C \leq \Delta \qquad \forall i \in \mathcal{A} \tag{4.62}$$

L'objectif (4.50) est de minimiser la durée de projet, sous contraintes d'antériorité (4.51) et sous contraintes de ressources agrégées par périodes (4.52). Les contraintes 4.53 (respectivement 4.54) permettent de calculer la date de début (respectivement de fin) d'exécution d'une activité. Les contraintes 4.55 traduisent la relation de partition au sein d'une période (cf. proposition 4.1). Les contraintes suivantes (4.56) à (4.58) assurent que les valeurs prises par les variables utiles dans les contraintes de ressources agrégées ($d_{i',i,\alpha}$, déduit à partir de $\lambda_{i',i,\alpha}$ et $\mu_{i',i,\alpha}$) soient cohérentes avec les valeurs prises par les variables modélisant les dates de début et de fin (ℓ_i^S, τ_i^S, ℓ_i^C et τ_i^C) grâce à des fonctions binaires $\mathbf{1}_X$, où X consiste en une comparaison entre deux variables entières à un décalage (valeur entière constante) près.

Linéarisation de la formulation initiale

La formulation initiale n'est pas linéaire, car les contraintes (4.57) et (4.58) font apparaître le produit d'une variable continue positive et d'une variable binaire. La proposition suivante présente un résultat connu en programmation linéaire en nombres entiers, qui permet de linéariser un tel produit.

Proposition 4.5 (Produit d'une variable continue positive et bornée avec une variable binaire)**.** *Soit $u \in \mathbb{R}$ une variable continue. Soit $v \in [0, \overline{v}]$ une variable continue, positive et bornée supérieurement par une constante. Soit $x \in \{0,1\}$ une variable binaire.*

$$u = v\,x \qquad \Leftrightarrow \qquad \begin{cases} u \geq 0 \\ u \leq v \\ u \geq v - \overline{v}(1-x) \\ u \leq \overline{v}\,x \end{cases}$$

Démonstration. La variable binaire x peut prendre soit la valeur 1 soit la valeur 0. Le tableau suivant indique, pour chaque contrainte, si elle est inactive ($\emptyset$) ou active

(expression dans laquelle la variable x est substituée par sa valeur).

	$x=0$	$x=1$
$u \geq v - \overline{v}(1-x)$	$\emptyset$	$u \geq v$
$u \leq \overline{v}\, x$	$u \leq 0$	$\emptyset$

□

Par ailleurs, afin d'obtenir une formulation linéaire, il est nécessaire de pouvoir modéliser les fonctions binaires 1_X uniquement à partir de contraintes linéaires. Ici, les expressions booléennes X s'expriment toutes sous la forme d'une comparaison entre deux variables entières en prenant en compte un décalage (entier également). À chaque fois, trois comparaisons sont réalisées : inégalité inférieure stricte, égalité, inégalité supérieure stricte. Nous proposons ci-après un ensemble de contraintes linéaires permettant de modéliser un tel comportement.

Proposition 4.6 (Comparaison de deux variables entières)**.** *Soit $u \in \{\underline{u}, \dots, \overline{u}\}$ et $v \in \{\underline{v}, \dots, \overline{v}\}$ des variables entières, bornées inférieurement et supérieurement par des constantes. Soit $c \in \mathbb{Z}$ un paramètre entier (constante). Soit $x \in \{0,1\}$, $y \in \{0,1\}$ et $z \in \{0,1\}$ des variables binaires.*

$$\begin{cases} (x=1) \Leftrightarrow (u < v+c) \\ (y=1) \Leftrightarrow (u = v+c) \\ (z=1) \Leftrightarrow (u > v+c) \end{cases} \quad \Leftrightarrow \quad \begin{cases} x+y+z & = 1 \\ u-v-c+1 & \leq (\overline{u}-\underline{v}-c+1)\,(1-x) \\ u-v-c & \leq (\overline{u}-\underline{v}-c) \quad z \\ v-u+c & \leq (\overline{v}-\underline{u}+c) \quad x \\ v-u+c+1 & \leq (\overline{v}-\underline{u}+c+1)\,(1-z) \end{cases}$$

Démonstration. Le triplet (x,y,z) peut prendre trois valeurs distinctes : $(1,0,0)$, $(0,1,0)$ ou $(0,0,1)$. Le tableau suivant indique, pour chaque contrainte, si elle est inactive ($\emptyset$) ou active (expression dans laquelle les variables x, y et z sont substituées par leurs valeurs).

	$x=1$ $y=0$ $z=0$	$x=0$ $y=1$ $z=0$	$x=0$ $y=0$ $z=1$
$u-v-c+1 \leq (\overline{u}-\underline{v}-c+1)\,(1-x)$	$u < v+c$	$\emptyset$	$\emptyset$
$u-v-c \leq (\overline{u}-\underline{v}-c) \quad z$	$u \leq v+c$	$u \leq v+c$	$\emptyset$
$v-u+c \leq (\overline{v}-\underline{u}+c) \quad x$	$\emptyset$	$u \geq v+c$	$u \geq v+c$
$v-u+c+1 \leq (\overline{v}-\underline{u}+c+1)\,(1-z)$	$\emptyset$	$\emptyset$	$u > v+c$

□

Renforcement de la formulation

La variable S_i peut être retirée du modèle, à condition de s'assurer que les variables ℓ_i^S, τ_i^S, ℓ_i^C et τ_i^C prennent des valeurs cohérentes, de sorte à respecter la durée de l'activité i (voir figure 4.8 page 87).

$$\tau_i^S + p_i + \tau_i^C = (\ell_i^C - \ell_i^S + 1)\Delta \qquad \forall i \in \mathcal{A} \tag{4.63}$$

De plus, il est possible de lier plus fortement ces quatre familles de variables afin de modéliser les contraintes d'antériorité, à partir des remarques suivantes. Pour tout $(i_1, i_2) \in E$, il existe toujours une représentation d'une solution telle que $\ell^C_{i_1} \leq \ell^S_{i_2}$. En cas d'égalité ($\ell^C_{i_1} = \ell^S_{i_2}$), la somme des offsets $\tau^C_{i_1}$ et $\tau^S_{i_2}$ doit être suffisante ($\geq \Delta$) afin d'éviter tout chevauchement ; sinon ($\ell^C_{i_1} < \ell^S_{i_2}$), aucune contrainte supplémentaire n'est nécessaire (du moins, pour satisfaire les relations d'antériorité).

$$\ell^S_{i_2} - \ell^C_{i_1} \geq 0 \qquad \forall (i_1, i_2) \in E \tag{4.64}$$

$$\tau^C_{i_1} + \tau^S_{i_2} \geq -(\ell^S_{i_2} - \ell^C_{i_1} - 1)\Delta \qquad \forall (i_1, i_2) \in E \tag{4.65}$$

Comme précédemment, il est possible d'introduire, pour chaque activité i telle que $p_i \bmod \Delta \neq 0$, une variable binaire π_i qui permet de déterminer le nombre de périodes intersectées par la fenêtre d'exécution de l'activité, ce qui revient ici à fixer l'écart entre ℓ^S_i et ℓ^C_i. Toutefois, l'intérêt est moindre pour cette formulation, de par sa compacité : cela ne permet pas de diminuer le nombre de variables soumises à une contrainte d'intégrité explicite.

$$\ell^C_i - \ell^S_i = \left\lfloor \frac{p_i}{\Delta} \right\rfloor + \pi_i \qquad \forall i \in \mathcal{A} \tag{4.66}$$

$$\pi_i \in \{0, 1\} \qquad \forall i \in \mathcal{A} \tag{4.67}$$

Remarque. Si une activité $i \in \mathcal{A}$ est telle que $p_i \bmod \Delta = 0$, il est alors possible de lier directement ℓ^S_i et ℓ^C_i.

$$\ell^C_i - \ell^S_i = \frac{p_i}{\Delta} \qquad \forall i \in \mathcal{A} \tag{4.68}$$

Chapitre 5

PARCPSP : méthodes approchées et résultats expérimentaux

Le PARCPSP étant un problème **NP**-difficile au sens fort, les approches de PLNE présentées dans le chapitre précédent rencontrent de fortes limitations pour des grandes instances du problème, surtout lorsque la durée des périodes diminue. Ce chapitre propose ainsi différentes heuristiques pour résoudre le problème, dont en premier lieu des algorithmes de liste inspirés des algorithmes parallèle et sériel du RCPSP. Ensuite, une heuristique itérative intégrant ces algorithmes de liste et un modèle de PLNE est proposée. Celle-ci est basée sur la résolution successive de PARCPSP, en faisant diminuer progressivement la durée des périodes, en espérant pouvoir exploiter les solutions réalisables pour une durée de périodes donnée, afin d'obtenir une solution réalisable lorsque la durée des périodes est inférieure, et ce, malgré les résultats négatifs d'approximation démontrés dans le 3. Des résultats expérimentaux concluent ce chapitre sur des instances de PARCPSP générées à partir de la PSPLIB, bibliothèque d'instances de problèmes pour le RCPSP (Kolisch et Sprecher 1996). Dans un premier temps, la qualité des heuristiques proposées est évaluée en comparaison de la première formulation de PLNE. Dans un second temps, les formulations de PLNE sont comparées entre elles, tant pour la qualité de la relaxation que pour la résolution en nombres entiers. Les résultats concernant les heuristiques et la comparaison avec la première formulation ont été publiés dans une revue (Morin, Artigues et Haït 2017b). Des résultats préliminaires de comparaison des formulations ont été présentés dans des colloques (Morin, Artigues et Haït 2017a ; Morin et al. 2018a,b).

5.1 Algorithmes de listes

Cette section est dédiée à la description d'algorithmes de listes pour le PARCPSP. Tout d'abord, la notion de liste de priorité est définie, et des algorithmes existants pour le RCPSP, parfois appelés Schedule Generation Schemes (SGS) dans la littérature, sont abordés (description et propriétés). Ensuite, une adaptation de ces algorithmes au cas du PARCPSP est proposée.

5.1.1 Algorithmes basés sur des listes de priorité pour le RCPSP

Liste de priorité

Définition 5.1 (Liste de priorité)**.** Une liste de priorité σ est une fonction bijective qui, à une position $j \in \{1, \dots, n\}$, associe une activité $i \in \mathcal{A}$. Cette fonction permet donc de définir un ordre total de l'ensemble des activités ($\mathcal{A}$) à partir de l'ordre croissant des positions ; cet ordre total doit être une extension de l'ordre partiel induit par les relations d'antériorité (E).

$$\begin{cases} \forall j_1 \in \{1, \dots, n\} \\ \forall j_2 \in \{1, \dots, n\} \end{cases} \qquad (\sigma(j_1), \sigma(j_2)) \in E \quad \Rightarrow \quad j_1 < j_2$$

Algorithme sériel

À partir d'une liste de priorité σ, l'algorithme sériel, parfois appelé Serial Schedule Generation Scheme (SSGS) dans la littérature, construit un ordonnancement en considérant chaque activité une par une dans l'ordre induit par σ, en faisant démarrer chaque activité au plus tôt (à partir de la date $t = 0$). Une version naïve du SSGS est proposée dans l'algorithme 5.1 ci-après.

```
1 pour j = 1 à n faire
2     i ← σ(j)
3     t ← 0
4     tant que l'activité i ne peut pas commencer à la date t faire
5         t ← t + 1
6     S_i ← t
```

Algorithme 5.1 : Algorithme sériel pour le RCPSP

Algorithme parallèle

À partir d'une liste de priorité σ, l'algorithme parallèle, parfois appelé Parallel Schedule Generation Scheme (PSGS) dans la littérature, construit un ordonnancement en essayant de faire commencer le plus grand nombre d'activités possible à un instant donné, avant de passer à l'instant suivant (à partir de la date $t = 0$). Une version naïve du PSGS est proposée dans l'algorithme 5.2 ci-après.

```
1 A' ← ∅                                        /* activités déjà ordonnancées */
2 t ← 0
3 tant que A' ≠ A faire
4     pour j = 1 à n faire
5         i ← σ(j)
6         si (i ∉ A') ∧ (L'activité i peut commencer à la date t) alors
7             S_i ← t
8             A' ← A' ∪ {i}
9     t ← t + 1
```

Algorithme 5.2 : Algorithme parallèle pour le RCPSP

Propriétés

Pour le RCPSP, il est possible de restreindre l'espace de recherche à l'ensemble de solutions dominantes caractérisé par les propriétés suivantes.

1. Toutes les dates de début d'exécution sont entières (puisque toutes les données numériques, i.e. durées d'exécution, capacités des ressources et consommations, sont supposées entières).
2. La date de début d'exécution pour toute activité coïncide soit avec la date $t = 0$, soit avec la date de fin d'exécution d'une autre activité. Ainsi, il existe au plus $n + 1$ instants auxquels un ou plusieurs événements de type début et/ou fin d'exécution se produisent.

Les versions naïves des algorithmes présentées précédemment n'ont pas une complexité polynomiale en fonction du nombre d'activités (n) et du nombre de ressources (m) ; en effet, leur complexité est également proportionnelle à la durée de la solution produite, à cause de l'incrément unitaire dans la boucle temporelle. Cependant, en utilisant la deuxième remarque, il est possible de remplacer les instructions $t \leftarrow t + 1$ par $t \leftarrow \text{next}(t)$, où next est une fonction qui renvoie, en temps constant, la prochaine date à laquelle (au moins) un événement se produit. Avec cette optimisation (sauvegarde en mémoire des dates de début/fin avec mise à jour du profil de consommation sur chaque ressource à chaque placement d'activité), la complexité du SSGS et du PSGS devient polynomiale.

De plus, il existe toujours une liste de priorité telle que la solution renvoyée par le SSGS appliqué sur cette liste est optimale (pour une preuve de ce résultat, voir par exemple Artigues et al. (2008), chapitre 6). Ce résultat ne s'applique pas au PSGS.

5.1.2 Adaptation au cas du PARCPSP

La suite de cette section est dédiée à l'adaptation des algorithmes SSGS et PSGS, initialement conçus pour le RCPSP, au cas du PARCPSP. En particulier, la seule différence entre ces deux problèmes réside dans la définition des contraintes de ressources. C'est pourquoi nous redéfinissons tout d'abord les tests de faisabilité par rapport aux contraintes de ressources agrégées sur des périodes de même durée, avant de passer aux pseudo-codes détaillés de ces algorithmes.

Calcul de la durée maximale d'exécution d'une activité dans une période en fonction des contraintes de ressources agrégées

Soit $\mathcal{A}' \subset \mathcal{A}$ un ensemble d'activités déjà ordonnancées, i.e. dont la date de début d'exécution a déjà été fixée. Soit $i \in \mathcal{A} \setminus \mathcal{A}'$ une nouvelle activité à ordonnancer.

Pour toute ressource $k \in \mathcal{R}$, dans toute période $\ell \in \mathbb{Z}$, les contraintes de ressources agrégées par périodes induisent une borne supérieure, notée $\overline{D_{i,\ell}}^{(k)}(\mathcal{A}')$, sur la durée d'exécution de l'activité i dans la période ℓ, i.e. $D_{i,\ell}(t)$, pour toute date de début d'exécution $t \in \mathbb{R}$, lorsque la consommation de l'activité i sur la ressource k est non nulle ($r_{i,k} > 0$). En effet, en ne considérant que les activités dans $\mathcal{A}' \cup \{i\}$, ces contraintes s'écrivent sous la forme :

$$\sum_{i' \in \mathcal{A}'} r_{i',k}\, D_{i',\ell}(S_{i'}) + r_{i,k}\, D_{i,\ell}(t) \leq b_k\, \Delta$$

D'où :

$$D_{i,\ell}(t) \leq \overline{D_{i,\ell}}^{(k)}(\mathcal{A}') = \frac{b_k\,\Delta - \sum_{i' \in \mathcal{A}'} r_{i',k}\,D_{i',\ell}(S_{i'})}{r_{i,k}}$$

Rappelons de plus que, par définition, $D_{i,\ell}(t) \leq \Delta$. Soit $\overline{D_{i,\ell}}(\mathcal{A}')$ la durée maximale d'exécution de l'activité i dans la période ℓ qui respecte les contraintes de ressources agrégées, simultanément pour tous les types de ressources.

$$\overline{D_{i,\ell}}(\mathcal{A}') = \min\left(\ \Delta\ ,\ \min_{k \in \mathcal{R}\,:\,r_{i,k}>0}\left(\overline{D_{i,\ell}}^{(k)}(\mathcal{A}')\right)\ \right)$$

Test de faisabilité de l'exécution d'une activité à date de début fixée en fonction des contraintes de ressources agrégées

Soit $\mathcal{A}' \subset \mathcal{A}$ un ensemble d'activités déjà ordonnancées, i.e. dont la date de début d'exécution a déjà été fixée. Soit $i \in \mathcal{A} \setminus \mathcal{A}'$ une nouvelle activité à ordonnancer, telle que $E_i^{\ominus} \subseteq \mathcal{A}'$, i.e. tous les prédecesseurs de i ont déjà été ordonnancés. Soit $t \in \mathbb{R}$ une date de début d'éxecution potentielle pour l'activité i.

- La date t est compatible avec les contraintes d'antériorité si et seulement si :

$$t \geq \max_{i' \in E_i^{\ominus}} (S_{i'} + p_{i'})$$

- La date t est compatible avec les contraintes de ressources agrégées si et seulement si $D_{i,\ell}(t) \leq \overline{D_{i,\ell}}(\mathcal{A}')$ dans toute période $\ell \in \mathbb{Z}$; il est suffisant de tester cette inégalité seulement dans les périodes $[(\ell-1)\Delta, \ell\Delta]$ dont l'intersection avec l'intervalle $[t, t+p_i]$ est non vide et non réduite à un point, i.e. $D_{i,\ell}(t) > 0$.

$$\bigwedge_{\ell \in \mathbb{Z}\,:\,D_{i,\ell}(t)>0} \left(D_{i,\ell}(t) \leq \overline{D_{i,\ell}}(\mathcal{A}')\right)$$

Algorithme sériel

Le pseudo-code de cette procédure, adaptée pour le PARCPSP, est proposé dans l'algorithme 5.3 page 95. Le principe reste similaire au SSGS pour le RCPSP : boucle principale (externe) sur les activités, boucle auxiliaire (interne) sur le temps.

Algorithme parallèle

Le pseudo-code de cette procédure, adaptée pour le PARCPSP, est proposé dans l'algorithme 5.4 page 96. Le principe reste similaire au PSGS pour le RCPSP : boucle principale sur le temps, boucle auxiliaire sur les activités.

Exemple

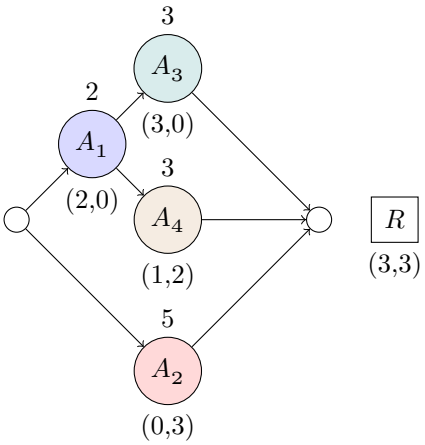
Afin d'illustrer le principe de fonctionnement de ces algorithmes, une description pas à pas de l'exécution du SSGS sur une instance avec $n = 4$ activités et $m = 2$ ressources est proposée ci-après (cf. figure 5.1 page 97). La liste de priorité considérée est $\sigma = \{1, 2, 3, 4\}$. Les contraintes de ressources sont agrégées sur des périodes de durée $\Delta = 3$.

```
   /* La notation D_{i,ℓ}(A') est définie page 94. */
   /* Le test réalisé ligne 12 est expliqué en détail page 94. */
1  A' ← ∅                                              /* activités déjà ordonnancées */
2  pour chaque j = 1 à n faire
3      i ← σ(j)                                        /* l'activité i est en position j dans σ */
4      t_min ← max_{i'∈E_i^⊖}(S_{i'} + p_{i'})         /* plus grande date de fin parmi les prédécesseurs */
5      ℓ ← ⌊t_min / Δ⌋
6      S_i ← max(t_min, ℓΔ − D_{i,ℓ}(A'))
7      si S_i = ℓΔ alors  /* D_{i,ℓ}(A') = 0 : l'activité i ne peut pas commencer dans la période ℓ */
8          répéter
9              ℓ ← ℓ + 1
10             S_i ← ℓΔ − D_{i,ℓ}(A')
11         jusqu'à S_i < ℓΔ    /* D_{i,ℓ}(A') > 0 : l'activité i peut peut-être commencer dans la période ℓ */
12     tant que l'activité i ne peut pas commencer à la date S_i faire
13         répéter
14             ℓ ← ℓ + 1
15             S_i ← ℓΔ − D_{i,ℓ}(A')
16         jusqu'à S_i < ℓΔ    /* D_{i,ℓ}(A') > 0 : l'activité i peut peut-être commencer dans la période ℓ */
17     si p_i ≤ ℓΔ − S_i alors    /* exécution de l'activité i dans une seule période */
18         S_i ← max(t_min, (ℓ − 1)Δ)
19     A' ← A' ∪ {i}                                   /* ajout de l'activité i dans l'ensemble des activités ordonnancées */
```

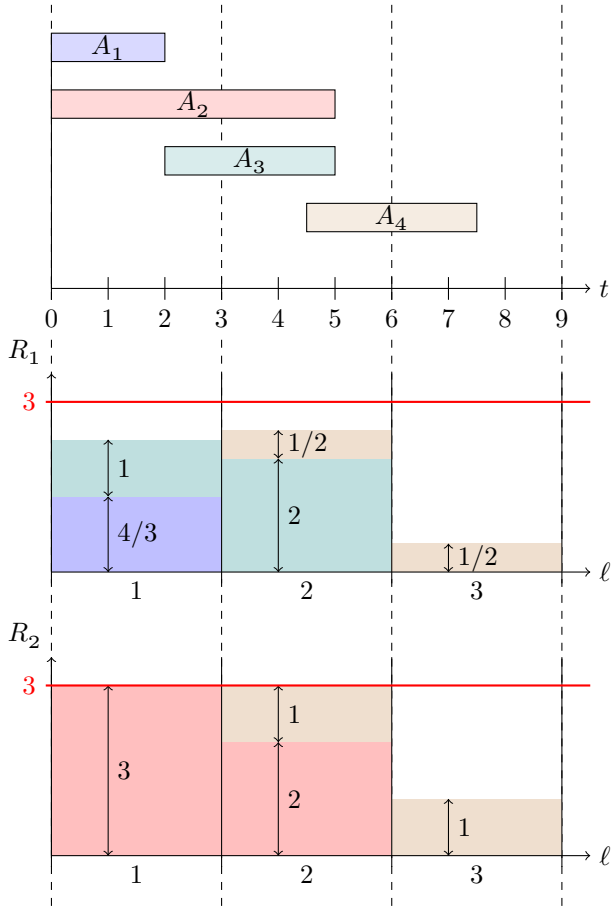
Algorithme 5.3 : Algorithme sériel pour le PARCPSP

```
   /* La notation D_{i,ℓ}(A') est définie page 94. */
   /* Le test réalisé ligne 11 est expliqué en détail page 94. */
1  A' ← ∅                                              /* activités déjà ordonnancées */
2  ℓ ← 1                                               /* démarrage dans la première période */
3  tant que A' ≠ A faire
4      i* ← 0                                          /* convention pour n'indiquer aucune activité */
5      t* ← ℓΔ                                         /* date de début potentielle pour l'activité i* */
6      pour j = 1 à n faire
7          i ← σ(j)                                    /* l'activité i est l'activité en position j dans σ */
8          si i ∉ A' et E_i^⊖ ⊆ A' alors
               /* l'activité i n'est pas encore ordonnancée */
               /* tous les prédécesseurs de l'activité i ont été ordonnancés */
9              t_min ← max_{i'∈E_i^⊖}(S_{i'} + p_{i'})  /* plus grande date de fin parmi les
               prédécesseurs */
10             t ← max(t_min, ℓΔ − D_{i,ℓ}(A'))
11             si t < t* et l'activité i peut commencer à la date t alors
12                 i* ← i
13                 t* ← t
14     si i* = 0 alors                                 /* plus aucune activité exécutable dans cette période */
15         ℓ ← ℓ + 1                                   /* passage à la période suivante */
16     sinon
17         si p_i ≤ ℓΔ − t* alors /* exécution de l'activité i dans une seule période */
18             t_min ← max_{i'∈E_i^⊖}(S_{i'} + p_{i'})  /* plus grande date de fin parmi les
               prédécesseurs */
19             S_i ← max(t_min, (ℓ − 1)Δ)
20         sinon
21             S_{i*} ← t*
22         A' ← A' ∪ {i*}                              /* ajout de l'activité i* dans l'ensemble des activités
           ordonnancées */
```

Algorithme 5.4 : Algorithme parallèle pour le PARCPSP



(a) Instance



(b) Solution générée par le SSGS et le PSGS

Fig. 5.1 : Exemple – Schedule Generation Scheme

1. L'activité 1 a une demande non-nulle seulement sur la première ressource. Celle-ci est exécutée à partir de la date $t = 0$; ainsi, sa fenêtre d'exécution $[0, 2]$ intersecte uniquelement la première période $[0, 3]$. Par définition, la consommation moyenne est égale à $r_{1,1} \frac{D_{1,1}(0)}{\Delta} = 2 \times \frac{2}{3} = \frac{4}{3}$.
2. L'activité 2 a une demande non-nulle seulement sur la deuxième ressource ; son exécution peut donc également démarrer au plus tôt à la date $t = 0$. Cette fois-ci, la fenêtre d'exécution $[0, 4]$ intersecte deux périodes : $[0, 3]$ et $[3, 6]$.
3. L'activité 3 doit être réalisée après l'activité 1 à cause d'une contrainte d'antériorité : ainsi, son exécution ne peut pas commencer avant la date $t = 2$. La contrainte sur la première ressource agrégée dans la première période n'est pas saturée ici, puisque la somme des consommations moyennes reste strictement inférieure à la capacité.
4. L'activité 4 doit elle aussi être réalisée après l'activité 1 (contrainte d'antériorité). Cependant, il est impossible de démarrer l'exécution de cette activité immédiatement après son prédécesseur à cause de la contrainte sur la deuxième ressource agrégée dans la première période (capacité déjà atteinte). De plus, à cause de la contrainte sur la même ressource mais dans la période suivante, cette activité ne peut pas être exécutée avant la date $t = 4, 5$.

Le lecteur pourra vérifier que cette solution, générée par le SSGS, est aussi celle générée par le PSGS.

Propriétés

Les propriétés de dominance établies pour le RCPSP ne sont malheureusement pas valides pour le PARCPSP.

- Il existe des instances pour lesquelles les solutions entières ne sont pas dominantes. C'est notamment le cas pour le premier exemple présenté dans le chapitre 2 (cf. section 2.2), pour lequel toute solution entière réalisable a une durée supérieure ou égale à 2, alors que la durée minimale atteignable vaut 1.
- Il existe des instances pour lesquelles les solutions avec au plus $n + 1$ événements ne sont pas dominantes. C'est notamment le cas pour le second exemple présenté dans le chapitre 2 (cf. section 2.2) : pour $\Delta = 2$, la solution optimale $S = (0, 1, 4)$ est telle qu'aucun événement de début/fin d'exécution ne coïncide avec un autre (cf. figure 2.10 page 32).

Par ailleurs, ces algorithmes sont construits de telle sorte que la première activité ordonnancée commence systématiquement au plus tôt, i.e. à la date $t = 0$; or l'ensemble des solutions qui commencent à la date $t = 0$ n'est pas dominant pour le PARCPSP. Par conséquent, s'il n'existe pas de solution optimale commençant à la date $t = 0$, alors il n'existe aucune liste de priorité qui permette de générer une solution optimale (ni avec le SSGS, ni avec le PSGS). Cette remarque s'applique au premier exemple présenté dans le chapitre 2. Cependant, le fait de devoir choisir une date de début systématique pour l'exécution du projet semble difficilement contournable. En effet, pour éviter d'avoir à faire ce choix arbitraire, il faudrait être en mesure de trouver une date de début de projet pour laquelle il existe (au moins) une solution optimale.

5.2 Schéma de résolution itératif

La méthode heuristique de résolution du PARCPSP proposée ci-après combine programmation mathématique (cf. modèles présentés dans le chapitre 4) et Schedule Generation Schemes (cf. algorithmes présentés précédemment dans ce chapitre).

La taille des modèles mathématiques est inversement proportionnelle à la durée des périodes Δ : en effet, plus Δ est petit, plus le nombre de périodes est élevé, et plus il est difficile de résoudre jusqu'à l'optimalité. Par ailleurs, les solveurs peuvent prendre en compte des informations supplémentaires afin d'accélérer la résolution, notamment lorsqu'on leur fournit des solutions réalisables et proches de l'optimum. L'idée directrice du schéma de résolution est de résoudre successivement plusieurs modèles en faisant décroître la durée des périodes au fur et à mesure, tout en propageant de l'information d'une itération à la suivante.

Soit $N \in \mathbb{N}^*$. Soit $\Delta_0, \Delta_1, \ldots, \Delta_N$ des durées de périodes (réels strictement positifs) triées par ordre décroissant ($\Delta_0 > \Delta_1 > \cdots > \Delta_N$). L'objectif est de trouver une solution optimale pour le PARCPSP défini avec des périodes de durée Δ_N (dernière itération). Le schéma de résolution est défini de la manière suivante.

1. Résoudre le problème défini avec des périodes de durée Δ_0 par PLNE
2. À chaque iteration $i = 1$ à N :
   (a) Générer un ensemble de solutions réalisables pour le problème défini avec des périodes de durée Δ_i, en réparant les meilleures solutions sauvegardées au cours des itérations précédentes
   (b) Résoudre le problème défini avec des périodes de durée Δ_i par PLNE, en accélérant potentiellement la résolution grâce aux bornes supérieures générées (démarrage à chaud du solveur, ou *warm start*)

Ce processus itératif est décrit de manière détaillé dans l'algorithme 5.5 page 100. Cet algorithme utilise des ensembles de solutions dont la taille est limitée, appelées *pool*, afin de garder uniquement un certain nombre de solutions, tout en conservant les meilleures à chaque insertion. Cette structure de données utilise les primitives suivantes :

- Pool(M)
  Constructeur ; le paramètre M indique le nombre maximal de solutions pouvant être stockées dans le *pool*.
- PoolGetAllHints(P)
  Itérateur en lecture seule ; permet de parcourir l'ensemble des éléments dans le *pool* P.
- PoolInsert(P, S)
  Permet d'ajouter des éléments dans un *pool*.
  - Si le *pool* P n'est pas rempli, i.e. n'a pas atteint sa taille maximale, la solution S est insérée dans P systématiquement.
  - Sinon, P est déjà rempli. Soit S' la solution de moins bonne qualité dans P.
    * Si S est meilleure que S', alors S' est retirée de P, tandis que S est ajoutée dans P.
    * Sinon, par définition de S', S est de moins bonne qualité que toutes les solutions dans P ; S est alors ignorée, i.e. le *pool* n'est pas modifié.

Le processus de réparation d'une solution S (première phase de la procédure RepairAndExploreFrom()) est effectué en deux étapes.

```
1  pour i = 0 à N faire
2      LB_i ← 0
3      P_i ← Pool(M)
4      si i = 0 alors
5          pour S ∈ Explore(Δ_i, ρ) faire
6              PoolInsert(P_i, S)
7      sinon
8          pour j = 0 à i − 1 faire
9              si Δ_j mod Δ_i ≠ 0 alors
                   /* Aucun lien a priori entre les problèmes définis avec des périodes de
                      durée Δ_j et Δ_i */
10                 pour S ∈ PoolGetAllHints(P_j) faire
11                     pour S' ∈ RepairAndExploreFrom(S, Δ_i, ρ) faire
12                         PoolInsert(P_i, S')
13             sinon
                   /* Le problème défini avec des périodes de durée Δ_j est une
                      relaxation de celui défini avec des périodes de durée Δ_i */
14                 LB_i = max(LB_i, LB_j)
15                 pour S ∈ PoolGetAllHints(P_j) faire
16                     pour S' ∈ ExploreFrom(S, Δ_i, ρ) faire
17                         PoolInsert(P_i, S')
18     UB_i, LB_i ← SolveMIPWarmStart(Δ_i, PoolGetAllHints(P_i), LB_i)
19     PoolInsert(P_i, UB_i)
```

Algorithme 5.5 : Iterative Solution Scheme

1. Transformation de S en une liste de priorité σ qui conserve l'ordre croissant des dates de début

$$\begin{cases} \forall j_1 \in \{1, \dots, n\} \\ \forall j_2 \in \{1, \dots, n\} \end{cases} \qquad S_{\sigma(j_1)} \leq S_{\sigma(j_2)} \quad \Rightarrow \quad j_1 \leq j_2$$

2. Application d'une heuristique (SSGS, PSGS) sur σ

De plus, afin d'apporter de la diversité dans le pool de solutions, de nouvelles solutions sont générées aléatoirement à partir de listes de priorités modifiées, obtenues en échangeant si possible les positions de deux activités successives avec une probabilité ρ (seconde et dernière phase de la procédure RepairAndExploreFrom(), et procédure ExploreFrom() ; cf. algorithme 5.6 ci-après).

```
1 pour chaque j = 2 à n faire
2     i_1 ← σ(j − 1)
3     i_2 ← σ(j)
4     si (rand(0, 1) ≤ ρ) ∧ ((i_1, i_2) ∉ E) alors
5         σ(j − 1) ← i_2
6         σ(j) ← i_1
```

Algorithme 5.6 : Randomisation aléatoire d'une liste de priorité

L'efficacité de cette méthode dépend de deux facteurs principaux : d'une part, le nombre d'itérations N ; d'autre part, les variations de la séquence décroissante $(\Delta_i)_{0 \leq i \leq N}$. Afin de propager une information d'aussi bonne qualité que possible tout au long du processus, le pas de décrémentation $(\Delta_i - \Delta_{i-1})$ doit diminuer d'itération en itération.

5.3 Évaluation expérimentale des heuristiques

5.3.1 Description des schémas de résolution

Nous comparons ici trois schémas de résolution.

1. La première méthode `[M1]` consiste à utiliser les algorithmes SSGS et PSGS sur un grand nombre de listes de priorité.
2. La seconde méthode `[M2]` utilise la première formulation indexée par les périodes, en fournissant au solveur des solutions réalisables (obtenues avec la méthode `[M1]`) pouvant être utilisées comme point de départ de la résolution (démarrage à chaud).
3. La troisième méthode `[M3]` correspond au schéma de résolution itératif présenté dans la section précédente. Plusieurs problèmes sont résolus les uns après les autres ; d'une itération à la suivante, la durée des périodes décroit, et l'ensemble des solutions initiales fournies au solveur est mis à jour en réparant les solutions obtenues lors des itérations précédentes.

Les instances avec 30 activités et 4 ressources de la PSPLIB ont été utilisées (480 instances). Le nombre maximal de listes de priorité considérées a été fixé à 1000 ; en partant de la liste de priorité qui correspond à la permutation identité (toujours valide pour les instances de la PSPLIB), de nouvelles listes sont générées en utilisant l'algorithme 5.6 page 101, avec une probabilité d'échange de deux indices consécutifs

(ρ) fixée à 20%. Pour la résolution des programmes linéaires en nombres entiers, le solveur utilisé est CPLEX 12.6.2 (IBM) ; pour chaque résolution, le temps de calcul maximal autorisé est de 3600 secondes.

5.3.2 Analyse des résultats

Les résultats pour l'ensemble des méthodes concernant l'écart (gap) à la meilleure borne supérieure (upper bound UB) ainsi que les temps de calcul (time) sont présentés dans le tableau 5.1. Le nombre de solutions optimales est également indiqué pour les méthodes `[M1]` and `[M2]`.

Concernant la méthode `[M3]`, il est important de noter que les résultats sont cumulatifs, c'est-à-dire que, pour la $r^{\text{ième}}$ ligne :

- Le problème résolu est le PARCPSP avec des périodes de durée Δ_r (la valeur de Δ_r apparaît dans la colonne la plus à gauche).
- Le processus itératif comporte r iterations : les valeurs des durées des périodes d'itération en itération sont $\Delta_1, \Delta_2, \cdots, \Delta_r$ (cf. lignes au-dessus).
- Les résultats prennent en compte l'ensemble du processus itératif (toutes les itérations).

En termes d'écart à la meilleure borne supérieure, les meilleures solutions sont toujours trouvées par la méthode `[M3]`. On peut remarquer que l'écart pour la méthode `[M1]` est assez grand, en particulier pour des grandes valeurs de Δ. Ce phénomène peut être expliqué par le fait que les solutions construites par les algorithmes de liste commencent invariablement à la date $t = 0$; or, cette classe de solutions n'est pas dominante pour le PARCPSP.

En termes de temps de calcul, les algorithmes de liste sont très rapides. De plus, la méthode `[M3]` est parfois plus rapide que la méthode `[M2]` : c'est le cas lorsque Δ est compris entre 4.5 et 2.5. La méthode `[M3]` parvient à prouver l'optimalité pour un plus grand nombre d'instances, mais le nombre de solutions optimales supplémentaires trouvées décroit avec Δ. Cela montre que le processus itératif permet de réduire le temps de calcul, en dépit du fait qu'il soit bien plus complexe, puisque plusieurs programmes linéaires en nombres entiers sont résolus au cours de celui-ci. Néanmoins, l'information obtenue en réparant des solutions est utile afin d'accélérer la résolution, jusqu'à un certain point. En effet, quand Δ devient trop petit, la qualité de l'information propagée tout au long du processus itératif est de moins en moins bonne. Cette caractéristique est cohérente avec les propriétés présentées dans le chapitre 3.

Dans la suite, une analyse plus détaillée des performances de la méthode `[M3]`, la plus performante, est réalisée sur un échantillon de 48 instances parmi celles de la PSPLIB (J30), afin d'étudier l'impact de la durée des périodes et les caractéristiques des instances. Les instances de la PSPLIB sont générées en fonction de trois paramètres. À chaque triplet de valeur possible correspond une classe de 10 instances dans la PSPLIB ; la première instance de chaque classe est sélectionnée.

1. Network Complexity (NC)
   Ce paramètre permet de mesurer la densité du graphe d'antériorité. Plus la valeur du paramètre est grande, plus le graphe est dense. Ce paramètre peut prendre 3 valeurs : 1.50, 1.80, 2.10.
2. Resource Factor (RF)

	[M1]		[M2]			[M3]		
Δ	Gap UB (%)	Time (s)	Gap UB (%)	Time (s)	# Opt (/480)	Gap UB (%)	Time (s)	# Opt (/480)
10	17.23	< 1	0	75	480	0	75	480
8	14.16	< 1	0	89	480	0	117	480
6	12.01	< 1	0.46	138	475	0	151	475
5	9.69	< 1	1.16	154	458	0	169	460
4	8.55	< 1	1.83	207	450	0	186	452
4.5	7.49	< 1	2.17	223	443	0	195	447
3	6.98	< 1	2.49	231	442	0	208	444
2.75	6.3	< 1	2.92	241	437	0	223	441
2.5	6.07	< 1	3.27	250	435	0	241	439
2.25	5.72	< 1	4.12	277	433	0	283	436
2	5.29	< 1	3.88	296	431	0	320	436
1.8	4.96	< 1	4.35	339	428	0	381	435
1.6	4.61	< 1	3.28	372	427	0	447	435
1.4	4.29	< 1	3.55	421	426	0	532	434
1.3	3.84	< 1	3.41	463	423	0	599	434
1.2	3.5	< 1	2.99	488	421	0	657	433
1.1	3.27	< 1	3.11	517	418	0	724	432
1	3.07	< 1	3.03	535	417	0	795	429

Tab. 5.1 : Comparaison des méthodes [M1], [M2] et [M3]

Ce paramètre permet de quantifier le nombre de ressources dont une activité a besoin au cours de son exécution. Ce paramètre peut prendre 4 valeurs, de 0.25 (1 ressource sur 4 utilisée) à 1 (4 ressources sur 4 utilisées).

3. Resource Strength (RS)
   Ce paramètre permet de comparer les capacités des ressources aux consommations des activités. Plus la valeur de ce paramètre est grande, plus le nombre d'activités pouvant être exécutées en parallèle est grand, car les capacités des ressources sont grandes par rapport aux consommations des activités. Ce paramètre peut prendre 4 valeurs : 0.75, 0.70, 0.50, 0.20.

Le tableau 5.2 indique, pour des durées de périodes $\Delta \in \{1, 2, 3, 4, 5\}$, et pour chaque classe regroupant les instances avec les mêmes paramètres, l'écart moyen à l'issue du branch-and-bound de la dernière itération, ainsi que le temps de calcul moyen, limité ici à 1800 secondes. Les résultats montrent que le temps de calcul et le gap augmentent lorsque NC diminue, RF augmente, et RS diminue. Cette tendance est similaire avec les résultats établis pour le RCPSP : les instances sont plus difficiles à résoudre lorsque le graphe d'antériorité est peu dense, les activités utilisent un nombre important de ressources, et les contraintes de ressources ne permettent d'exécuter simultanément qu'un petit nombre d'activités. De plus, lorsque les contraintes de ressources sont très agrégées (i.e. Δ est grand), l'impact des trois paramètres est plus faible : cela peut s'expliquer par le fait qu'il est plus facile de résoudre le programme linéaire en nombres entiers, notamment parce que le modèle, indexé par les périodes, est de plus petite taille. Cependant, les instances pour lesquelles RS vaut 0.20 restent difficiles : pour $\Delta = 5$, l'optimalité n'est atteinte que 9 fois sur 12, avec un temps moyen de 602 secondes.

Par ailleurs, il est également pertinent d'évaluer, en fonction des mêmes indicateurs, le gain en termes de durée du projet obtenu en agrégeant les contraintes de ressources par période. Le tableau 5.3 indique quel est l'écart entre l'optimum du RCPSP et l'optimum du PARCPSP. Parmi la même sélection de 48 instances, sont prises en compte celles pour lesquelles l'optimalité a été prouvée à la fin du processus itératif. Ici, on peut remarquer que le seul paramètre ayant un impact observable sur l'écart RCPSP/PARCPSP est RS. En particulier, l'optimum du RCPSP et l'optimum du PARCPSP sont identiques lorsque RS = 1, ce qui est cohérent avec le fait que tout se passe comme s'il n'y avait pas de contraintes de ressources. Par ailleurs, plus la durée des périodes sur lesquelles les contraintes de ressources sont agrégées est grande, plus cet écart est grand. Ce phénomène est particulièrement marqué pour les instances telles que RS = 0.20, puisque l'écart atteint 20% pour $\Delta = 5$. Il est également intéressant de remarquer que l'écart est déjà significatif lorsque les contraintes de ressources sont agrégées sur des périodes de durée unitaire ($\Delta = 1$) : même sur de petites périodes, le fait d'autoriser les activités à démarrer à n'importe quelle instant dans une période, tout en limitant non plus la consommation totale instantanée mais la consommation totale moyenne par période, permet une diminution non négligeable de la durée du projet.

5.4 Comparaison expérimentale des formulations PLNE

Cette section est dédiée à la comparaison des deux formulations indexées par période présentées dans le chapitre 4 (cf. sections 4.2.1 et 4.2.2 pages 74 et 79).

	$\Delta = 1$			$\Delta = 2$			$\Delta = 3$			$\Delta = 4$			$\Delta = 5$		
	# Opt	Gap (%)	Time (s)	# Opt	Gap (%)	Time (s)	# Opt	Gap (%)	Time (s)	# Opt	Gap (%)	Time (s)	# Opt	Gap (%)	Time (s)
NC=1.50 (16)	14	3.10	266	14	2.48	289	14	1.87	236	14	1.19	238	14	0.46	229
NC=1.80 (16)	14	1.95	325	14	2.01	273	14	1.70	248	14	0.31	233	16	0.02	94
NC=2.10 (16)	15	1.67	179	15	1.00	187	15	0.52	208	15	0.90	150	15	0.28	138
RF=1.00 (12)	9	6.38	569	9	4.48	510	9	3.07	478	9	2.37	458	10	0.82	387
RF=0.75 (12)	10	2.56	366	10	2.82	390	10	2.36	419	10	0.83	347	11	0.19	207
RF=0.50 (12)	12	0.02	64	12	0.01	90	12	0.02	19	12	0.01	19	12	0.01	15
RF=0.25 (12)	12	0.00	28	12	0.01	9	12	0.01	7	12	0.00	5	12	0.00	5
RS=1.00 (12)	12	0.00	15	12	0.00	4	12	0.00	2	12	0.00	2	12	0.00	1
RS=0.70 (12)	12	0.01	32	12	0.00	7	12	0.00	4	12	0.00	2	12	0.00	2
RS=0.50 (12)	12	0.01	143	12	0.01	133	12	0.02	32	12	0.02	11	12	0.01	8
RS=0.20 (12)	7	8.94	837	7	7.30	855	7	5.45	885	7	3.19	814	9	1.01	602

Tab. 5.2 : Méthode `[M3]` – Impact des paramètres NC, RF et RS

	$\Delta = 1$	$\Delta = 2$	$\Delta = 3$	$\Delta = 4$	$\Delta = 5$
All (48)	2.31	3.83	4.62	5.09	6.02
NC=1.50 (16)	2.44	4.22	4.98	5.22	5.24
NC=1.80 (16)	1.95	3.23	4.16	4.77	7.29
NC=2.10 (16)	2.51	4.02	4.72	5.26	5.38
RF=1.00 (12)	1.46	2.28	2.76	2.99	4.66
RF=0.75 (12)	1.77	2.50	3.06	3.57	5.45
RF=0.50 (12)	3.31	5.91	6.97	7.67	8.00
RF=0.25 (12)	2.39	4.01	4.98	5.34	5.69
RS=1.00 (12)	0.00	0.00	0.00	0.00	0.00
RS=0.70 (12)	0.75	0.82	0.82	0.82	0.82
RS=0.50 (12)	3.58	5.16	5.74	6.10	6.39
RS=0.20 (12)	6.75	13.27	17.15	19.38	20.47

Tab. 5.3 : Écart moyen (%) entre le RCPSP et le PARCPSP

5.4.1 Comparaison des relaxations linéaires

Les relaxations linéaires des deux formulations ont été testées dans les deux configurations suivantes.

1. Test du modèle de base
   (a) Pour la formulation 1, les contraintes (4.2) à (4.18) et (4.19) (démarrage du projet dans la première période) sont prises en compte.
   (b) Pour la formulation 2, les contraintes (4.27) à (4.39) et (4.41) sont prises en compte, sauf les contraintes d'antériorité agrégées (4.28) qui sont remplacées par les contraintes d'antériorité désagrégées (4.40).

   Les résultats pour cette configuration sont répertoriés dans le tableau 5.4a.
2. Test du modèle renforcé avec les variables π_i et les contraintes associées
   (a) Pour la formulation 1, les contraintes (4.18) (intégrité explicite des variables binaires permettant d'identifier dans quelle période une activité termine son exécution) sont supprimées, et les contraintes (4.20) à (4.26) sont ajoutées.
   (b) Pour la formulation 2, les contraintes (4.38) sont supprimées, et les contraintes (4.42) à (4.49) sont ajoutées.

Avant de lancer la résolution, le nombre de périodes utilisées pour construire le modèle est réduit autant que possible de manière heuristique. Les algorithmes SSGS et PSGS sont appliqués sur 1000 listes de priorité générées à partir de la permutation identité et l'algorithme 5.6 page 101 ($\rho = 20\%$). Pour la résolution des programmes linéaires (relaxations d'une part, en nombres entiers d'autre part), le solveur utilisé est CPLEX 12.6.2 (IBM).

Les tests ont été réalisés sur les 480 instances de projets de la PSPLIB comportant 30 activités et 4 ressources (J30), avec des périodes de durée $\Delta \in \{1, 2, 3, 4, 5\}$. Les résultats sont regroupés dans le tableau 5.4 page 107. Les indicateurs présentés pour chaque valeur de Δ sont détaillés ci-après.

- Time F1 (exprimé en secondes)
  Temps de calcul utilisé par la première formulation indexée par les périodes

Δ	Time F1 (s)	Time F2 (s)	# (CP < LB1) (/480)	# (CP < LB2) (/480)	# (LB1 < LB2) (/480)	Gap (%)	Gap (LB1 < LB2) (%)	Gap max (%)
5	0,051	0,191	0	0	0	0	N/A	0
4	0,075	0,339	0	0	0	0	N/A	0
3	0,146	0,666	0	0	0	0	N/A	0
2	0,288	1,890	0	0	0	0	N/A	0
1	0,688	11,015	0	0	0	0	N/A	0

(a) Sans les variables π_i

Δ	Time F1 (s)	Time F2 (s)	# (CP < LB1) (/480)	# (CP < LB2) (/480)	# (LB1 < LB2) (/480)	Gap (%)	Gap (LB1 < LB2) (%)	Gap max (%)
5	0,058	0,157	0	11	11	0,07	3,04	12,38
4	0,056	0,225	0	14	14	0,08	2,59	13,42
3	0,072	0,281	0	33	33	0,35	5,10	20,39
2	0,121	0,300	1	63	63	0,92	7,01	25,62
1	0,326	0,269	5	159	159	2,18	6,59	34,38

(b) Avec les variables π_i

Tab. 5.4 : Comparaison des relaxations des formulations indexées par période

- Time F2 (exprimé en secondes)
  Temps de calcul utilisé par la seconde formulation indexée par les périodes
- # (CP < LB1)
  Nombre d'instances pour lesquelles la valeur de la relaxation de la première formulation est strictement meilleure que la longueur du chemin critique (critical path CP) dans le graphe d'antériorité.
- # (CP < LB2)
  Nombre d'instances pour lesquelles la valeur de la relaxation de la seconde formulation est strictement meilleure que la longueur du chemin critique (critical path CP) dans le graphe d'antériorité.
- # (LB1 < LB2)
  Nombre d'instances pour lesquelles la valeur de la relaxation de la seconde formulation est strictement meilleure que celle de la première formulation.
  Rappelons que, d'après la proposition 4.3 page 82, la relation LB 1 $\leq$ LB 2 est toujours valide.
- Gap (exprimé en %)
  Écart relatif moyen entre les valeurs des deux relaxations ; ici, les 480 instances sont prises en compte.
- Gap (LB 1 < LB 2) (exprimé en %)
  Écart relatif moyen entre les valeurs des deux relaxations ; ici, seules les instances pour lesquelles la seconde formulation domine strictement la première sont prises en compte.
- Gap max (exprimé en %)
  Écart relatif maximal entre les valeurs des deux relaxations

Pour les deux formulations, l'introduction de la variable π_i permet de diminuer considérablement les temps de calcul. Ceci est particulièrement vrai pour la seconde formulation lorsque la durée des périodes Δ diminue.

Concernant les configurations sans la variable π_i (cf. tableau 5.4a page 107), les valeurs des deux relaxations ne sont jamais meilleures que la longueur du chemin critique.

En revanche, pour les configurations avec la variable π_i (cf. tableau 5.4b page 107), la valeur de la première formulation est meilleure que cette borne dans quelques cas lorsque Δ est petit ; cependant, cette valeur est toujours dominée par la valeur de la seconde formulation, et même dominée strictement sur un nombre d'instances de plus en plus en grand lorsque Δ diminue. Les écarts relatif moyens et maximaux entre les deux modèles décroient également avec Δ.

La première formulation tourne plus rapidement que la seconde, sauf avec les variables π_i lorsque $\Delta = 1$. Ce phénomène est sans doute dû au fait que, dans ce cas, toutes les durées (entières) d'exécution des activités sont des multiples de Δ. Les variables π_i ne sont donc jamais introduites, mais les contraintes (4.49) sont ajoutées, permettant ainsi de diminuer le nombre de variables (relations d'égalité liant les variables $\lambda_{i,\ell}$ et $\mu_{i,\ell}$).

5.4.2 Comparaison des formulations (en nombres entiers)

Ici, seuls les résultats pour la deuxième configuration sont présentés. En effet, les résultats des tests des relaxations linéaires montrent que les formulations renforcées avec les variables π_i (et les contraintes associées) sont meilleures à la fois en termes de temps de calcul et de qualité de la borne inférieure. Les test ont été réalisés

sur une sélection de 48 instances de projets de la PSPLIB comportant 30 activités et 4 ressources (J30), également avec des périodes de durée $\Delta \in \{1, 2, 3, 4, 5\}$. Les résultats sont regroupés dans le tableau 5.5 page 110. Les indicateurs présentés pour chaque valeur de Δ sont détaillés ci-après.

- Time (exprimé en secondes)
  Temps de calcul moyen sur l'ensemble des 48 instances, limité à 3600 secondes par instance
- # Opt
  Nombre d'instances résolues jusqu'à l'optimalité
- # UB Fail
  Nombre d'instances pour lesquelles le solveur n'est pas parvenu à trouver une solution en nombres entiers dans le temps de calcul imparti pour la résolution.
- Gap (exprimé en %)
  Écart relatif moyen entre la borne supérieure (meilleure solution entière trouvée) et la borne inférieure à la fin du *branch-and-bound* ; ici, seules les instances n'ayant pas été résolues jusqu'à l'optimalité et pour lesquelles le solveur a trouvé une borne supérieure (solution en nombres entiers) sont prises en compte.

Pour les deux formulations, de $\Delta = 5$ à 2, le temps de calcul augmente ; en revanche, pour $\Delta = 1$, le temps de calcul diminue. La même hypothèse évoquée précédemment pourrait expliquer ce phénomène : pour $\Delta = 1$, la variable π_i n'est jamais introduite, puisque les durées des activités (entières) sont toutes des multiples de Δ. Ainsi, des contraintes d'égalité ((4.26) pour la première formulation, (4.48) et (4.49) pour la seconde) permettent de réduire le nombre de variables ; aussi, les quatre familles de contraintes liées à la variable π_i ne sont jamais introduites dans les modèles ((4.22) à (4.25) pour la première formulation, (4.44) à (4.47) pour la seconde). Les instances pour lesquelles $\Delta = 1$ semblent plus faciles à résoudre que celles pour lesquelles $\Delta = 2$; cela est corroboré par le fait que, pour la seconde formulation, le solveur ne trouve pas de borne supérieure (solution entière) pour une instance lorsque $\Delta = 2$.

De $\Delta = 5$ à 3, la première formulation est plus lente que la seconde, mais plus rapide pour $\Delta = 2$ et 1. De manière similaire, pour les valeurs de Δ les plus grandes, plus d'instances sont résolues jusqu'à l'optimalité par la seconde formulation que par la première ; c'est le contraire pour les valeurs de Δ les plus petites. Cela peut être lié à la taille des modèles, qui décroit avec Δ, et plus rapidement pour la seconde formulation, car les contraintes d'antériorité agrégées (4.28) ont été remplacées par leur version désagrégée (4.40). Néanmoins, en termes d'écart relatif à la fin du *branch-and-bound*, la seconde formulation est en moyenne toujours meilleure que la première.

Δ	Time (s)	# Opt (/48)	# UB Fail (/48)	Gap (%)
5	137,18	47	0	3,87
4	258,29	46	0	2,57
3	362,34	44	0	5,70
2	381,71	44	0	7,61
1	316,31	44	0	13,33

(a) Formulation 1 avec les variables π_i

Δ	Time (s)	# Opt (/48)	# UB Fail (/48)	Gap (%)
5	66,84	48	0	N/A
4	220,96	47	0	0,62
3	327,34	44	0	3,27
2	414,60	43	1	5,45
1	379,35	43	0	7,29

(b) Formulation 2 avec les variables π_i

Tab. 5.5 : Comparaison des formulations indexées par période

Conclusion

La littérature consacrée à la gestion de projet est d'une immense diversité. Le problème d'ordonnancement de projet sous contraintes de ressources (RCPSP) est le problème standard dans ce domaine, qui convient particulièrement à un niveau de décision opérationnel. De nombreuses variantes ont été proposées afin de prendre en compte des contraintes spécifiques. Certaines d'entre elles sont plus adaptées à un niveau de décision tactique, car elles permettent une gestion plus souple de la durée des activités et de leurs consommations en ressources.

Le problème introduit dans cette thèse, intitulé problème d'ordonnancement de projet sous contraintes de ressources avec agrégation périodique (PARCPSP), se situe entre ces deux niveaux de décision : en effet, l'aspect temporel est géré de manière fine, notamment en ce qui concerne les relations d'antériorité entre activités, tandis que l'évaluation de la consommation des ressources est discrétisée et agrégée sur des périodes de même durée. Contrairement à d'autres problèmes qui autorisent plus de flexibilité concernant l'exécution d'une activité au cours du temps, tels que le Rough Cut Capacity Planning (RCCP, introduit par Hans 2001) ou le RCPSP avec intensité variable (RCPSVP, introduit par Kis 2005), les durées opératoires sont ici connues, et les consommations des activités sur les ressources sont constantes au cours du temps. De plus, les dates de début et de fin d'activités sont modélisées comme des variables continues, ce qui permet d'assurer l'existence de solutions réalisables, contrairement à d'autres problèmes où le temps est discrétisé, comme le RCPSP avec ressources partiellement renouvelables (RCPSP/π introduit par Böttcher et al. 1999).

Nous avons établi que le PARCPSP est un problème **NP**-difficile : il s'agit donc d'un problème combinatoire difficile. De plus, nous avons montré que de nombreuses règles de dominance courantes pour d'autres problèmes classiques en ordonnancement ne sont pas valides pour ce problème. En particulier, les solutions entières ne sont pas dominantes, et la faisabilité d'une solution peut être altérée lorsqu'elle est décalée dans le temps.

Nous avons ainsi été amenés à définir les concepts de faisabilité locale, caractérisant les solutions pouvant devenir réalisables après translation, et de faisabilité globale, caractérisant les solutions qui sont réalisables et qui le restent après translation, qui peuvent être vérifiés en temps polynomial. Grâce à ces concepts, nous avons pu établir des liens forts entre le PARCPSP et le RCPSP, le premier étant une relaxation du second, mais qui fournit une approximation arbitrairement mauvaise dans le pire cas. Nous avons également montré qu'une variation de la durée des périodes sur lesquelles les contraintes de ressources sont agrégées peut complètement changer l'espace des solutions.

Ensuite, nous avons proposé des modèles exacts de programmation linéaire en nombres entiers pour ce problème. Deux modèles originaux à temps mixte (variables continues pour les dates de début et de fin, et discrétisation périodique) ont été proposés et comparés théoriquement en termes de qualité de leur relaxation linéaire, le second dominant strictement le premier. Un modèle compact a également été présenté. Ces trois modèles ont pu être renforcés en exploitant la non-préemption des activités ainsi que la non-variabilité des durées d'exécution.

Enfin, des approches heuristiques dédiées au PARCPSP ont été proposées. Des algorithmes classiques de la littérature à base de listes de priorité ont été adaptés afin de prendre en compte l'agrégation des contraintes de ressource. Par ailleurs, un schéma de résolution itératif combinant les méthodes précédentes et considérant un horizon temporel de moins en moins agrégé a aussi été implémenté. Les résultats expérimentaux montrent que le problème est difficile à résoudre, même pour des instances de taille modeste avec seulement 30 tâches.

De nombreuses perspectives de recherches sont envisageables afin d'approfondir l'étude du PARCPSP. Un premier axe de recherche concerne les méthodes de résolution du problème.

D'un point de vue théorique, la formulation compacte n'a pas une bonne relaxation linéaire, car les techniques utilisées pour linéariser le modèle requièrent l'utilisation de contraintes de type *big-*$\mathcal{M}$. Néanmoins, d'un point de vue computationnel, il serait intéressant d'évaluer ses performances par rapport aux formulations indexées par période.

La mise en place de schémas de décomposition pour le PARCPSP est aussi une piste intéressante. Par exemple, des formulations étendues proposées pour le RCPSP pourraient être adaptées à ce problème. L'une des difficultés à surmonter concerne l'aspect temporel continu du PARCPSP, qui impacte également les contraintes de ressources agrégées par période. Nous avons présenté dans (Morin et al. 2018a) un travail préliminaire sur une telle décomposition, dont l'exploitation peut constituer une piste de recherche à court terme (les dates de début sont représentées dans le problème maître non pas par une variable continue, mais prennent leurs valeurs dans un sous-ensemble fini et discret, ces valeurs étant générées dynamiquement par le sous-problème).

Concernant les heuristiques, les algorithmes de listes adaptés pour le PARCPSP ont une complexité pseudo-polynomiale, puisqu'elle dépend du nombre de périodes utilisées par la solution construite. Le pseudo-code de ces procédures pourrait être retravaillé afin d'atteindre une complexité au pire cas polynomiale. De plus, la question de l'existence d'un algorithme de liste avec la garantie de l'existence d'une liste de priorité menant à une solution optimale est toujours ouverte.

Par ailleurs, des méthodes de résolution basées sur d'autres paradigmes comme la programmation par contraintes pourraient aussi être étudiées. Là encore, l'aspect continu du problème pourrait être perçu comme un obstacle, ou alors comme une opportunité d'étudier la version discrète du problème. Des méthodes hybrides mêlant programmation par contraintes et programmation linéaire en nombres entiers, comme celle implémentée par Artigues et al. (2009) pourraient également être testées, en distinguant deux échelles de temps : l'une détaillée pour l'exécution des activités en termes de dates et de relations de séquence, l'autre agrégée pour la consommation des ressources.

Concernant la subdivision de l'horizon temporel en périodes, seul le cas uniforme a été traité dans ce manuscrit. Une perspective de recherche concerne donc la généralisation des propriétés et des modèles aux cas de subdivisions temporelles non uniformes, et de les transposer dans des approches de résolution basées sur des horizons glissants.

La fonction objectif considérée dans cette thèse est la durée du projet à minimiser. D'autres objectifs pourraient être considérés, par exemple liés à l'utilisation des ressources, comme le lissage de la consommation sur l'horizon de planification, ou la minimisation des coûts engendrés par le recours à des ressources externes.

Plusieurs variantes du PARCPSP se prêtent à des approches multi-niveaux, notamment des modèles intégrés mêlant planification et ordonnancement. Pour cela, il est nécessaire d'apporter de la flexibilité au niveau de décision tactique. Cela devient possible en considérant par exemple des activités de durée variable, inversement proportionnelle à leur consommation instantanée (puissance) en ressources, la complétion d'une activité étant dans ce cas déterminée par une quantité d'énergie à fournir.

Enfin, les modèles considérés ici sont déterministes : ils ne permettent pas de gérer les aléas survenant au cours du projet, liés à la durée des activités ou la disponibilité des ressources. Par conséquent, des méthodes à base de scénarios issues d'une décomposition hiérarchique de la décision avec une politique robuste au niveau tactique (planification) et réactive au niveau opérationnel (ordonnancement) pourraient ainsi être mises en œuvre.

Bibliographie

Alfieri, A., Tolio, T. et Urgo, M. (2011). « A Project Scheduling Approach to Production Planning with Feeding Precedence Relations ». In : *International Journal of Production Research* 49.4, p. 995–1020.

Allen, J. F. (1981). « An Interval-Based Representation of Temporal Knowledge ». In : *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 1*. IJCAI'81. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., p. 221–226.

Artigues, C. (2013). « A Note on Time-Indexed Formulations for the Resource-Constrained Project Scheduling Problem ».

Artigues, C. (2017). « On the Strength of Time-Indexed Formulations for the Resource-Constrained Project Scheduling Problem ». In : *Operations Research Letters* 45.2, p. 154–159.

Artigues, C., Brucker, P. et al. (2013). « A Note on "Event-Based MILP Models for Resource-Constrained Project Scheduling Problems" ». In : *Computers & Operations Research* 40.4, p. 1060–1063.

Artigues, C., Demassey, S. et Néron, E., éds. (2008). *Resource-Constrained Project Scheduling : Models, Algorithms, Extensions and Applications.* London, UK : ISTE.

Artigues, C., Lopez, P. et Haït, A. (2013). « The Energy Scheduling Problem : Industrial Case-Study and Constraint Propagation Techniques ». In : *International Journal of Production Economics* 143.1, p. 13–23.

Artigues, C. et al. (2009). « Solving an Integrated Employee Timetabling and Job-Shop Scheduling Problem via Hybrid Branch-and-Bound ». In : *Computers & Operations Research*. Constraint Programming 36.8, p. 2330–2340.

Artigues, C. et al. (2015). « Nouvelles relaxations basées sur la programmation linéaire en nombres entiers pour le problème d'ordonnancement de projet sous contraintes de ressources ». In : 16ème congrès annuel de la société Française de Recherche Opérationnelle et d'Aide à la Décision ROADEF 2015. Marseille, p. 2.

Baptiste, P. et al. (2005). *Gestion de Production et Ressources Humaines : Méthodes de Planification Dans Les Systèmes Productifs*. Sous la dir. de P. Baptiste et al. Montreal, Canada : Presses Internationales Polytechniques. 308 p.

Baydoun, G. et al. (2016). « A Rough-Cut Capacity Planning Model with Overlapping ». In : *OR Spectrum* 38.2, p. 335–364.

Bianco, L. et Caramia, M. (2012). « Minimizing the Completion Time of a Project under Resource Constraints and Feeding Precedence Relations : An Exact Algorithm ». In : *4OR* 10.4, p. 361–377.

Bianco, L. et Caramia, M. (2013). « A New Formulation for the Project Scheduling Problem under Limited Resources ». In : *Flexible Services and Manufacturing Journal* 25.1, p. 6–24.

Blazewicz, J., Lenstra, J. K. et Kan, A. H. G. R. (1983). « Scheduling Subject to Resource Constraints : Classification and Complexity ». In : *Discrete Applied Mathematics* 5.1, p. 11–24.

Blazewicz, J. et al. (1986). *Scheduling under Resource Constraints : Deterministic Models*. Annals of operations research 7. OCLC : 16883877. Basel : Baltzer. 359 p.

Böttcher, J. et al. (1999). « Project Scheduling Under Partially Renewable Resource Constraints ». In : *Management Science* 45.4, p. 543–559.

Browning, T. R. et Yassine, A. A. (2010). « Resource-Constrained Multi-Project Scheduling : Priority Rule Performance Revisited ». In : *International Journal of Production Economics* 126.2, p. 212–228.

Carvalho, A. N., Oliveira, F. et Scavarda, L. F. (2015). « Tactical Capacity Planning in a Real-World ETO Industry Case : An Action Research ». In : *International Journal of Production Economics* 167, p. 187–203.

Castro, P. M., Harjunkoski, I. et Grossmann, I. E. (2011). « Optimal Scheduling of Continuous Plants with Energy Constraints ». In : *Computers & Chemical Engineering* 35.2, p. 372–387.

Castro, P., Harjunkoski, I. et Grossmann, I. (2009). « New Continuous-Time Scheduling Formulation for Continuous Plants under Variable Electricity Cost ». In : *Industrial & Engineering Chemistry Research - IND ENG CHEM RES* 48.

Cherkaoui, K. et al. (2017). « Proactive Tactical Planning Approach for Large Scale Engineering and Construction Projects ». In : *The Journal of Modern Project Management* 5.1.

Christofides, N., Alvarez-Valdes, R. et Tamarit, J. M. (1987). « Project Scheduling with Resource Constraints : A Branch and Bound Approach ». In : *European Journal of Operational Research* 29.3, p. 262–273.

Coelho, J. et Valadares, L. T. (2002). « Comparative Analysis on Approximation Algorithms for the Resource Constrained Project Scheduling Problem ». In : *Eighth International Workshop on Project Management and Schduling*.

Drótos, M. et Kis, T. (2011). « Resource Leveling in a Machine Environment ». In : *European Journal of Operational Research* 212.1, p. 12–21.

Dumbravă, Ş. (2011). « Aggregated Models Technique for Integrating Planning and Scheduling of Production Tasks ». In : *International Journal of Modern Manufacturing Technologies* 3.1, p. 39–44.

Dupin, N. (2015). « Modélisation et Résolution de Grands Problèmes Stochastiques Combinatoires : Application à La Gestion de Production d'électricité ». thesis. Lille 1.

Egri, P. et al. (2004). « Project-Oriented Approach to Production Planning and Scheduling in Make to Order Manufacturing ». In : *Production Systems and Information Engineering* 2, p. 23–36.

Even, G. et al. (2008). « Algorithms for Capacitated Rectangle Stabbing and Lot Sizing with Joint Set-up Costs ». In : *ACM Trans. Algorithms* 4.3, 34 :1–34 :17.

Fischetti, M., Sartor, G. et Zanette, A. (2015). « MIP-and-Refine Matheuristic for Smart Grid Energy Management ». In : *International Transactions in Operational Research* 22.1, p. 49–59.

Floudas, C. A. et Lin, X. (2004). « Continuous-Time versus Discrete-Time Approaches for Scheduling of Chemical Processes : A Review ». In : *Computers & Chemical Engineering* 28.11, p. 2109–2129.

Fündeling, C. .-.-U. et Trautmann, N. (2010). « A Priority-Rule Method for Project Scheduling with Work-Content Constraints ». In : *European Journal of Operational Research* 203.3, p. 568–574.

Gademann, N. et Schutten, M. (2005). « Linear-Programming-Based Heuristics for Project Capacity Planning ». In : *IIE Transactions* 37.2, p. 153–165.

Gahm, C. et al. (2016). « Energy-Efficient Scheduling in Manufacturing Companies : A Review and Research Framework ». In : *European Journal of Operational Research* 248.3, p. 744–757.

Gajic, D. et al. (2017). « Implementation of an Integrated Production and Electricity Optimization System in Melt Shop ». In : *Journal of Cleaner Production. Sustainable Development of Energy, Water and Environmental Systems* 155, p. 39–46.

Garey, M. R. et Johnson, D. S. (1978). « " Strong " NP-Completeness Results : Motivation, Examples, and Implications ». In : *J. ACM* 25.3, p. 499–508.

Garey, M. et Johnson, D. (1975). « Complexity Results for Multiprocessor Scheduling under Resource Constraints ». In : *SIAM Journal on Computing* 4.4, p. 397–411.

Garey, M. R. et Johnson, D. S. (1990). *Computers and Intractability ; A Guide to the Theory of NP-Completeness*. New York, NY, USA : W. H. Freeman & Co.

Gélinas, S. (2004). « Problèmes d'ordonnancement ». PhD Thesis. Montreal, Canada : École polytechnique.

Haït, A. et Baydoun, G. (2012). « A New Event-Based MILP Model for the Resource-Constrained Project Scheduling Problem with Variable Intensity Activities ». In : *The IEEE International Conference on Industrial Engineering and Engineering Management*.

Hait, A. et Artigues, C. (2011). « An Hybrid CP/MILP Method for Scheduling with Energy Costs ». In : *European Journal of Industrial Engineering* 5.4, p. 471–489.

Haït, A. et Artigues, C. (2011). « On Electrical Load Tracking Scheduling for a Steel Plant ». In : *Computers & Chemical Engineering* 35.12, p. 3044–3047.

Hans, E. W. (2001). « Resource Loading by Branch-and-Price Techniques ». OCLC : 933757075. Enschede : Twente Univ. Press. 151 p.

Hartmann, S. et Kolisch, R. (2000). « Experimental Evaluation of State-of-the-Art Heuristics for the Resource-Constrained Project Scheduling Problem ». In : *European Journal of Operational Research* 127.2, p. 394–407.

Kis, T. et Kovács, A. (2012). « A Cutting Plane Approach for Integrated Planning and Scheduling ». In : *Computers & Operations Research* 39.2, p. 320–327.

Kis, T. (2005). « A Branch-and-Cut Algorithm for Scheduling of Projects with Variable-Intensity Activities ». In : *Mathematical Programming* 103.3, p. 515–539.

Kis, T. (2006). « RCPS with Variable Intensity Activities and Feeding Precedence Constraints ». In : *Perspectives in Modern Project Scheduling*. Sous la dir. de J. Józefowska et J. Weglarz. International Series in Operations Research & Management Science. Boston, MA : Springer US, p. 105–129.

Kis, T. et Drótos, M. (2017). « Hard Planning and Scheduling Problems in the Digital Factory ». In : *Math for the Digital Factory*. Sous la dir. de L. Ghezzi, D.

Hömberg et C. Landry. Mathematics in Industry. Cham : Springer International Publishing, p. 3–19.

Kolisch, R. et Hartmann, S. (1999). « Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem : Classification and Computational Analysis ». In : *Project Scheduling : Recent Models, Algorithms and Applications*. Sous la dir. de J. Węglarz. International Series in Operations Research & Management Science. Boston, MA : Springer US, p. 147–178.

Kolisch, R. et Hartmann, S. (2006). « Experimental Investigation of Heuristics for Resource-Constrained Project Scheduling : An Update ». In : *European Journal of Operational Research* 174.1, p. 23–37.

Kolisch, R. et Sprecher, A. (1996). « PSPLIB - A Project Scheduling Problem Library ». In : *European Journal of Operational Research* 96.1, p. 205–216.

Koné, O. et al. (2011). « Event-Based MILP Models for Resource-Constrained Project Scheduling Problems ». In : *Computers & Operations Research*. Project Management and Scheduling 38.1, p. 3–13.

Kovacs, A. (2005). « Novel Models and Algorithms for Integrated Production Planning and Scheduling ». Budapest.

Kovács, A. et Kis, T. (2011). « Constraint Programming Approach to a Bilevel Scheduling Problem ». In : *Constraints* 16.3, p. 317–340.

L'Heureux, G., Gamache, M. et Soumis, F. (2013). « Mixed Integer Programming Model for Short Term Planning in Open-Pit Mines ». In : *Mining Technology* 122.2, p. 101–109.

Maravelias, C. T. et Sung, C. (2009). « Integration of Production Planning and Scheduling : Overview, Challenges and Opportunities ». In : *Computers & Chemical Engineering*. FOCAPO 2008 – Selected Papers from the Fifth International Conference on Foundations of Computer-Aided Process Operations 33.12, p. 1919–1930.

Márkus, A. et al. (2003). « Project Scheduling Approach to Production Planning ». In : *CIRP Annals* 52.1, p. 359–362.

Masmoudi, M., Hans, E. W. et Haït, A. (2016). « Tactical Project Planning under Uncertainty : Fuzzy Approach ». In : *European Journal of Industrial Engineering* 10, p. 301–317.

Mestry, S., Damodaran, P. et Chen, C.-S. (2011). « A Branch and Price Solution Approach for Order Acceptance and Capacity Planning in Make-to-Order Operations ». In : *European Journal of Operational Research* 211.3, p. 480–495.

Mingozzi, A. et al. (1998). « An Exact Algorithm for the Resource-Constrained Project Scheduling Problem Based on a New Mathematical Formulation ». In : *Management Science* 44.5, p. 714–729.

Möhring, R. H. et al. (2001). « On Project Scheduling with Irregular Starting Time Costs ». In : *Operations Research Letters* 28.4, p. 149–154.

Möhring, R. H. et al. (2003). « Solving Project Scheduling Problems by Minimum Cut Computations ». In : *Management Science* 49.3, p. 330–350.

Monostori, L. et al. (2010). « Digital Enterprise Solution for Integrated Production Planning and Control ». In : *Computers in Industry*. Integration and Information in Networked Enterprises 61.2, p. 112–126.

Morin, P.-A., Artigues, C. et Haït, A. (2016a). « A New Relaxation of the Resource-Constrained Project Scheduling Problem ». In : 15th International Conference

on Project Management and Scheduling PMS 2016. Valencia, Spain : ADEIT Fundación Universidad Empresa, p. 9–12.

Morin, P.-A., Artigues, C. et Haït, A. (2016b). « A New Relaxation of the Resource-Constrained Project Scheduling Problem ». In : 29th Conference of the European Chapter on Combinatorial Optimization ECCO 2016. Budapest, Hungary.

Morin, P.-A., Artigues, C. et Haït, A. (2017a). « A New Mixed Time Framework for the Periodically Aggregated Resource-Constrained Project Scheduling Problem ». In : 13th Workshop on Models and Algorithms for Planning and Scheduling Problems MAPSP 2017. Seeon Abbey, Germany.

Morin, P.-A., Artigues, C. et Haït, A. (2017b). « Periodically Aggregated Resource Constrained Project Scheduling Problem ». In : *European Journal of Industrial Engineering* 11.6, p. 792–817.

Morin, P.-A., Artigues, C. et Haït, A. (2018a). « A Column Generation Scheme for the Periodically Aggregated Resource-Constrained Project Scheduling Problem ». In : 15th International Conference on Project Management and Scheduling PMS 2018. Rome, Italia, p. 165–168.

Morin, P.-A., Artigues, C. et Haït, A. (2018b). « Propriétés Structurelles et Formulations Linéaires Pour Le Problème d'ordonnancement de Projet Sous Contraintes de Ressources Avec Agrégation Périodique ». In : 19ème Édition Du Congrès Annuel de La Société Française de Recherche Opérationnelle et d'Aide à La Décision ROADEF 2018. Lorient, France.

Morin, P.-A., Artigues, C., Haït, A. et Ngueveu, S. U. (2017). « Comparaison de Diverses Formulations Pour Le RCPSP ». In : 18ème Édition Du Congrès Annuel de La Société Française de Recherche Opérationnelle et d'Aide à La Décision ROADEF 2017. Metz, France.

Morin, P.-A., Artigues, C., Hait, A. et al. (2017). *Structural Properties and Complexity of the Periodically Aggregated Resource-Constrained Project Scheduling Problem*. Rapp. tech. Toulouse : LAAS-CNRS.

Moukrim, A., Quilliot, A. et Toussaint, H. (2015). « An Effective Branch-and-Price Algorithm for the Preemptive Resource Constrained Project Scheduling Problem Based on Minimal Interval Order Enumeration ». In : *European Journal of Operational Research* 244.2, p. 360–368.

Naber, A. (2017). « Resource-Constrained Project Scheduling with Flexible Resource Profiles in Continuous Time ». In : *Computers & Operations Research* 84, p. 33–45.

Naber, A. et Kolisch, R. (2014). « MIP Models for Resource-Constrained Project Scheduling with Flexible Resource Profiles ». In : *European Journal of Operational Research* 239.2, p. 335–348.

Nattaf, M. et al. (2016). « Polyhedral Results and Valid Inequalities for the Continuous Energy-Constrained Scheduling Problem ».

Neumann, K., Schwindt, C. et Zimmermann, J. (2003). *Project Scheduling with Time Windows and Scarce Resources : Temporal and Resource-Constrained Project Scheduling with Regular and Nonregular Objective Functions ; with 60 Tables.* 2. ed. OCLC : 249470451. Berlin : Springer. 385 p.

Ngueveu, S. U., Artigues, C. et Lopez, P. (2016). « Scheduling under a Non-Reversible Energy Source : An Application of Piecewise Linear Bounding of Non-Linear Demand/Cost Functions ». In : *Discrete Applied Mathematics* 208, p. 98–113.

Nobibon, F. T. et al. (2013). « Resource Loading : Applications and Complexity Analysis ». In : p. 15–19.

Okubo, H. et al. (2015). « Project Scheduling under Partially Renewable Resources and Resource Consumption during Setup Operations ». In : *Computers & Industrial Engineering* 83, p. 91–99.

Paul, M. et Knust, S. (2015). « A Classification Scheme for Integrated Staff Rostering and Scheduling Problems ». In : *RAIRO - Operations Research* 49.2, p. 393–412.

Pinedo, M. L. (2016). *Scheduling : Theory, Algorithms, and Systems*. Springer.

Pritsker, A. a. B. et Watters, L. J. (1968). *A Zero-One Programming Approach to Scheduling with Limited Resources*. RM-5561-PR. RAND CORP SANTA MONICA CALIF.

Pritsker, A. A. B., Watters, L. J. et Wolfe, P. M. (1969). « Multiproject Scheduling with Limited Resources : A Zero-One Programming Approach ». In : *Manage. Sci.* 16.1, p. 93–108.

Riedler, M. et al. (2017). « An Iterative Time-Bucket Refinement Algorithm for a High-Resolution Resource-Constrained Project Scheduling Problem : M. Riedler et Al. » In : *International Transactions in Operational Research*.

Schwindt, C. et Zimmermann, J., éds. (2015a). *Handbook on Project Management and Scheduling. Vol. 1*. International handbooks on information systems. OCLC : 931819844. Cham : Springer. 663 p.

Schwindt, C. et Zimmermann, J., éds. (2015b). *Handbook on Project Management and Scheduling. Vol. 2*. International handbooks on information systems. OCLC : 931819846. Cham : Springer. 667 p.

Silvente, J. et al. (2015). « Improved Time Representation Model for the Simultaneous Energy Supply and Demand Management in Microgrids ». In : *Energy* 87, p. 615–627.

Sourd, F. (2005). « Optimal Timing of a Sequence of Tasks with General Completion Costs ». In : *European Journal of Operational Research* 165.1, p. 82–96.

Sousa, J. P. (1989). « Time Indexed Formulations of Non-Preemptive Single-Machine Scheduling Problems ». Louvain-la-Neuve, Belgium : Université Catholique de Louvain.

Souza, C. C. de et Wolsey, L. A. (1997). *Scheduling Projects with Labour Constraints*.

Tesch, A. (2015). *Compact MIP Models for the Resource-Constrained Project Scheduling Problem*.

Tolio, T. et Urgo, M. (2007). « A Rolling Horizon Approach to Plan Outsourcing in Manufacturing-to-Order Environments Affected by Uncertainty ». In : *CIRP Annals* 56.1, p. 487–490.

Toussaint, H. (2013). *Introduction au Branch Cut and Price et au solveur SCIP (Solving Constraint Integer Programs)*. LIMOS, p. 52.

Valls, V., Quintanilla, S. et Ballestín, F. (2003). « Resource-Constrained Project Scheduling : A Critical Activity Reordering Heuristic ». In : *European Journal of Operational Research*. Sequencing and Scheduling 149.2, p. 282–301.

Van de Vonder, S. (2006). « Proactive-Reactive Procedures for Robust Project Scheduling ».

Van de Vonder, S. et al. (2005). « The Use of Buffers in Project Management : The Trade-off between Stability and Makespan ». In : *International Journal of Production Economics* 97.2, p. 227–240.

Váncza, J., Kis, T. et Kovács, A. (2004). « Aggregation - the Key to Integrating Production Planning and Scheduling ». In : *CIRP Annals* 53.1, p. 377–380.
Watters, L. J. (1967). « Reduction of Integer Polynomial Programming Problems to Zero-One Linear Programming Problems ». In : *Operations Research* 15.6, p. 1171–1174.

AUTEUR

Pierre-Antoine MORIN

TITRE

Planification et ordonnancement de projets sous contraintes de ressources complexes

DIRECTEURS DE THÈSE

Alain HAÏT et Christian ARTIGUES

LIEU ET DATE DE LA SOUTENANCE

ISAE-SUPAERO, Toulouse, le 06/12/2018

RÉSUMÉ

La structure de projet se retrouve dans de nombreux contextes de l'industrie et des services. Il s'agit de réaliser un ensemble d'activités pouvant être connectées par des liens logiques de séquence (antériorité), en faisant appel à des ressources disponibles en quantité limitée. L'objectif est la minimisation d'un critère généralement lié à la durée ou au coût du projet. La plupart des problèmes d'ordonnancement de projet dans la littérature considèrent une unité de temps commune pour la détermination des dates d'exécution des activités et pour l'évaluation instantanée du respect des capacités des ressources qu'elles utilisent. Or, s'il est souvent nécessaire en pratique d'obtenir un calendrier détaillé des plages d'exécution des activités, l'utilisation des ressources peut être évaluée sur un horizon plus agrégé, comme par exemple les quarts de travail des employés. Dans cette thèse, un nouveau modèle intégrant ces deux échelles de temps est présenté afin de définir le problème d'ordonnancement de projet avec agrégation périodique des contraintes de ressources (PARCPSP). Ce problème est étudié du point de vue de la théorie de la complexité et des propriétés structurelles sont établies, mettant notamment en évidence des différences majeures avec le problème classique d'ordonnancement de projet sous contraintes de ressources (RCPSP). De ces propriétés sont dérivées des formulations exactes basées sur la programmation linéaire en nombres entiers, comparées en termes de qualité de la relaxation linéaire. Par ailleurs, plusieurs heuristiques, telles que des algorithmes de liste, ou une méthode approchée basée sur une résolution itérative qui exploite différentes échelles de temps, sont proposées. Les résultats expérimentaux montrent l'intérêt de ces différentes méthodes et illustrent la difficulté du problème.

MOTS-CLÉS

Planification ; ordonnancement ; projet ; optimisation combinatoire ;
programmation linéaire en nombres entiers.

DISCIPLINE ADMINISTRATIVE

Informatique / Optimisation

UNITÉS DE RECHERCHE

- Département d'Ingénierie des Systèmes Complexes (DISC)
  ISAE-SUPAERO – Institut Supérieur de l'Aéronautique et de l'Espace
  10 avenue Edouard Belin, BP 54032, 31055 Toulouse Cedex 4
- Équipe Recherche Opérationnelle, Optimisation Combinatoire et Contraintes (ROC)
  LAAS-CNRS – Laboratoire d'Analyse et d'Architecture des Systèmes
  7 avenue du Colonel Roche, BP 54200, 31031 Toulouse Cedex 4