
# 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** *18/10/2016* **par :**

MARGAUX NATTAF

## Ordonnancement sous contraintes d'énergie

**JURY**

| | | |
|---|---|---|
| CHRISTIAN ARTIGUES | Directeur de recherche au CNRS, LAAS-CNRS | Directeur de thèse |
| PHILIPPE BAPTISTE | Directeur de recherche au CNRS | Rapporteur |
| CYRIL BRIAND | Professeur, Université Toulouse 3-Paul Sabatier | Examinateur |
| TAMÁS KIS | Research fellow, MTA SZTAKI, Budapest, Hongrie | Examinateur |
| PHILIPPE LABORIE | Principal Scientist, IBM, Paris | Examinateur |
| PIERRE LOPEZ | Directeur de recherche au CNRS, LAAS-CNRS | Directeur de thèse |
| ALAIN QUILLIOT | Professeur, Université Blaise Pascal, Clermont-Ferrand | Examinateur |
| CLAUDE-GUY QUIMPER | Professeur agrégé, Université Laval, Québec, Canada | Rapporteur |

**École doctorale et spécialité :**
*EDSYS : Informatique 4200018*
**Unité de Recherche :**
*Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS)*
**Directeur(s) de Thèse :**
*Pierre LOPEZ* et *Christian ARTIGUES*
**Rapporteurs :**
*Philippe BAPTISTE* et *Claude-Guy QUIMPER*

# Remerciements

Je tiens d'abord à remercier mes deux directeurs de thèse, Pierre Lopez et Christian Artigues, pour m'avoir donné la chance de faire cette thèse avec eux. Son bon déroulement est grandement dû à leurs qualités scientifiques et humaines. Merci de m'avoir si bien encadrée et supportée pendant ces trois dernières années.

Je voudrais aussi remercier Alain Quillot pour l'honneur qu'il m'a fait en présidant le jury de ma soutenance.

Je remercie aussi Claude Guy Quimper et Philippe Baptiste d'avoir accepté d'être rapporteur de cette thèse et pour leur conseil d'experts sur mes travaux.

Je tiens aussi à remercier Philippe Laborie, Tamás Kis et Cyril Briand pour avoir accepter de participer à mon jury en temps qu'examinateur mais aussi pour l'intérêt qu'ils ont montré dans ce travail.

J'adresse également mes sincères remerciements à tous les membres de l'équipe ROC et à son personnel administratif grâce à qui ces trois années de thèse ont pu se dérouler dans de si bonnes conditions.

Merci à tous les gens qui m'ont permis de décompresser dans les moments difficiles (et les moments faciles aussi). Grâce à eux, ces trois ans de vie toulousaine ont été un plaisir (tellement que je reste encore un peu).

Merci aussi aux doctorants du Laboratoire l'Informatique et de Robotique et de Micro-électronique de Montpellier qui m'ont aidé sur les problèmes que j'ai rencontré lors de cette thèse et qui m'ont toujours gardé une petite place pour mes visite.

Enfin, merci à tous ceux qui, de près ou de loin, ont contribué à l'aboutissement de cette thèse. Á tous ceux qui m'ont supporté et à tous ceux qui n'ont jamais douté que j'y arriverai.

# Table des matières

**I Présentation du problème** **17**

**1 Ordonnancement, ressources cumulatives et énergie** **19**

1.1 Ordonnancement et contraintes de ressources . . . . . 19

1.1.1 L'ordonnancement . . . . . 19

1.1.2 Contraintes de ressources . . . . . 21

1.2 L'ordonnancement sous contraintes énergétiques . . . . . 26

1.2.1 Définition du problème . . . . . 26

1.2.2 Autres modélisations des activités à profil variable . . . . . 30

1.2.3 Propriétés du CECSP . . . . . 32

**II Programmation par contraintes** **41**

**2 Programmation par contraintes et ordonnancement cumulatif** **43**

2.1 La programmation par contraintes . . . . . 43

2.1.1 Problème de satisfaction de contraintes . . . . . 43

2.1.2 Exploration de l'espace de recherche . . . . . 44

2.1.3 Détection d'incohérence et Filtrage . . . . . 46

2.2 L'ordonnancement cumulatif . . . . . 48

2.2.1 L'ordonnancement en programmation par contraintes . . . . . 48

2.2.2 La contrainte cumulative . . . . . 49

2.2.3 Les filtrages de la contrainte cumulative . . . . . 50

**3 Propagation de contraintes pour le CECSP** **67**

3.1 Algorithmes de filtrage basés sur le Time-Table . . . . . 67

3.1.1 Le Time-Table . . . 68
3.1.2 Le Time-Table disjonctif . . . 69
3.1.3 Le Time-Table basé sur les flots . . . 73
3.2 Algorithme de filtrage du raisonnement énergétique . . . 75
3.2.1 Algorithme de vérification . . . 76
3.2.2 Les ajustements de bornes . . . 81
3.2.3 Caractérisation des intervalles d'intérêt . . . 85

**III Programmation linéaire en nombres entiers 103**

**4 Programmation linéaire et ordonnancement de projet 105**
4.1 La programmation linéaire en nombres entiers . . . 105
4.1.1 La programmation linéaire . . . 105
4.1.2 Contrainte d'intégrité . . . 106
4.1.3 Techniques de résolution . . . 106
4.2 Application au RCPSP . . . 107
4.2.1 Modèles indexés par le temps . . . 108
4.2.2 Modèles à événements . . . 109

**5 Programmation linéaire pour le CECSP et renforcement des modèles 115**
5.1 Modèles de programmation linéaire mixte pour le CECSP . . . 115
5.1.1 Modèle indexé par le temps . . . 115
5.1.2 Modèles à événements . . . 119
5.2 Renforcement des modèles . . . 126
5.2.1 Amélioration des modèles indexés par le temps . . . 127
5.2.2 Amélioration des modèles à événements . . . 129

**IV Implémentations et Expérimentations 141**

**6 Implémentations et Expérimentations 143**
6.1 Génération des instances et pré-traitement . . . 143
6.1.1 Instances du CECSP . . . 143

6.1.2 Instances et pré-calcul des fenêtres de temps pour le RCPSP . . . . . 145
6.2 Performances de la Programmation Linéaire Mixte . . . . . 146
6.2.1 Modèle indexé par le temps . . . . . 146
6.2.2 Modèles à événements . . . . . 147
6.2.3 Comparaison des différentes approches . . . . . 152
6.3 Performances de la Programmation Par Contraintes . . . . . 153
6.3.1 Cadre des expérimentations . . . . . 153
6.3.2 Raisonnement énergétique . . . . . 154
6.3.3 Raisonnements basés sur le Time-Table . . . . . 156
6.3.4 Comparaison des différentes approches . . . . . 157

**A Programmation linéaire mixte et programmation par contraintes pour un problème d'ordonnancement à contraintes énergétiques 173**

A.1 Introduction . . . . . 173
A.2 Modèle de programmation par contraintes . . . . . 174
A.2.1 Raisonnement énergétique . . . . . 175
A.3 Modèle de programmation linéaire en nombres entiers . . . . . 176
A.3.1 Modèle . . . . . 176
A.4 Inégalités valides basées sur le raisonnement énergétique . . . . . 177
A.5 Résultats expérimentaux . . . . . 177

**B A batch sizing and scheduling problem on parallel machines with different speeds, maintenance operations, setup times and energy costs 181**

B.1 Introduction . . . . . 181
B.2 The industrial scheduling problem . . . . . 182
B.3 Related work and modeling issues . . . . . 185
B.4 A continuous time event-based mixed-integer linear programming formulation . . . . . 187
B.5 A simplified model and a matheuristic . . . . . 190
B.6 Computational experiments . . . . . 192
B.7 Conclusion . . . . . 194

# Table des figures

1.1 Exemple d'instance pour le RCPSP. . . . . 23
1.2 Exemple de solutions réalisables pour le RCPSP. . . . . 23
1.3 Exemple de solutions réalisables pour le CuSP. . . . . 24
1.4 Exemple d'approximation d'une fonction de rendement non linéaire par une fonction affine et par une fonction concave et affine par morceaux. . . . . 28
1.5 Différentes formes de fonction d'allocation de ressource pour le CECSP. . . . . 29
1.6 Exemple d'une instance pour le CECSP. . . . . 30
1.7 Exemple de solution pour le CECSP. . . . . 30
1.8 Exemple de solution non-entière pour une instance du CECSP à données entières. . . 33

2.1 Exemple de parcours d'un arbre de recherche. . . . . 45
2.2 Exemple de parcours d'un arbre de recherche avec détection d'incohérences. . . . . 47
2.3 Exemple de parcours d'un arbre de recherche avec fitrage des domaines. . . . . 47
2.4 Partie obligatoire d'une activité $i$ pour le CuSP. . . . . 51
2.5 Exemple de profil obligatoire d'une ressource cumulative. . . . . 52
2.6 Règle d'ajustement du Time-Table pour le CuSP. . . . . 52
2.7 Raisonnement disjonctif pour le CuSP. . . . . 53
2.8 Calcul de l'énergie minimale dans un intervalle $[t_1, t_2[$ pour le CuSP. . . . . 56
2.9 Évolution de la fonction de consommation minimale en fonction de $t_2$ pour le CuSP. . 58
2.10 Ajustement de borne du raisonnement énergétique pour le CuSP. . . . . 60
2.11 Intervalle minimum de superposition d'une activité pour le CuSP. . . . . 62
2.12 Illustration de la règle 2.5 pour le CuSP. . . . . 63
2.13 Illustration du Time-Table disjonctif dans le cadre du CuSP. . . . . 64

3.1 Partie obligatoire d'une activité $i$ pour le CECSP. . . . . 69

3.2 Fonction $f_j(b_j(t))$. . . . . 70

3.3 Raisonnement disjonctif pour le CECSP. . . . . 70

3.4 Intervalle minimum de superposition d'une activité dans le cadre du CECSP. . . . . 71

3.5 Raisonnement disjonctif restreint pour le CECSP. . . . . 72

3.6 Illustration du Time-Table disjonctif pour le CECSP. . . . . 73

3.7 Les différentes configurations menant à une consommation minimale à l'intérieur de $[t_1, t_2[$ pour le CECSP. . . . . 77

3.8 Consommation de ressource dans $[t_1, t_2[$ pour le CECSP avec fonction de rendement affine. . . . . 80

3.9 Consommation de ressource dans $[t_1, t_2[$ pour le CECSP avec fonction de rendement concave et affine par morceaux. . . . . 82

3.10 Une solution réalisable du CECSP. . . . . 84

3.11 Les différentes expressions de $\underline{w}(i, t_1, t_2)$ en fonction des paramètres du problème CECSP. 89

3.12 Les différentes expressions de $\underline{b}(i, t_1, t_2)$ en fonction des paramètres du problème CECSP. 92

3.13 Intervalles d'intérêt du raisonnement énergétique du CECSP dans le cas où $est_i < t_1 \leq D'_{t_1}$. . . . . 95

3.14 Variation de $\underline{b}_{RS}(i, t_1, t_2)$ en fonction de $t_2$. . . . . 99

6.1 Création du graphe de l'algorithme de séparation pour les inégalités de non préemption 148

6.2 Procédure de branchement du l'algorithme hybride du CECSP. . . . . 154

A.1 Exemple de solution du modèle PPC. . . . . 175

B.1 Optimal solution of a simple instance. . . . . 185

B.2 Two batches of a job on the same machine. . . . . 186

# Liste des tableaux

1.1 Exemple d'une instance du CuSP. . . . . 24

1.2 Principales différences entre les extensions des problèmes cumulatifs et le CECSP. . . 32

3.1 Intervalles d'intérêt pour l'algorithme de vérification du raisonnement énergétique pour le CECSP. . . . . 94

3.2 Intervalles d'intérêt pour les ajustements du raisonnement énergétique pour le CECSP(placement à droite). . . . . 98

3.3 Intervalles d'intérêt pour les ajustements du raisonnement énergétique pour le CECSP(placement à gauche). . . . . 100

6.1 Résultats du PLNE indexé par le temps du CECSP avec et sans coupes énergétiques. 147

6.2 Résultats du modèle Start/End pour le CECSP(Famille 1). . . . . 150

6.3 Résultats du modèle On/Off pour le CECSP avec différentes combinaisons de coupes (Famille 4). . . . . 150

6.4 Résultats du modèle On/Off pour le CECSP avec différentes combinaisons de coupes (Famille 1). . . . . 151

6.5 Résultats du modèle On/Off pour le RCPSP avec différentes combinaisons de coupes. 151

6.6 Comparaison du modèle On/Off et du modèle indexé par le temps pour le CECSP. . . 152

6.7 Comparaison des trois modèles de PLNE du CECSP sans fonction objectif. . . . . 153

6.8 Comparaison des méthodes de calcul des intervalles d'intéret du raisonnement énergétique. 155

6.9 Résultats du raisonnement énergétique dans la méthode de branchement hybride pour le CECSP. . . . . 156

6.10 Résultats du Time-Table basé sur les flots dans la méthode de branchement hybride pour le CECSP. . . . . 157

6.11 Comparaison des résultats du modèle On/Off et de ceux de la méthode de branchement hybride pour le CECSP. . . . . 158

6.12 Résultats de la méthode de branchement arborescente sur les instances avec fonctions de rendement concaves et affines par morceaux. . . . . 158

A.1 Résultats du PLNE avec et sans inégalités valides : ER et DEF resp. (TL 1000s) . . . 178

A.2 Résultats du modèle PPC . . . . . 178

B.1 Setup times between batches. . . . . 192

B.2 Machine speeds for each lot. . . . . 193

B.3 Results of experiments for (MILP2+MILP3). . . . . 194

B.4 Detailed results of experiments for (MILP2+MILP3). . . . . 197

# Introduction

De nos jours, de nombreuses tâches, auparavant fastidieuses, ont vu leur complexité largement diminuée grâce à la mise en place d'outils informatiques permettant leur traitement. Ces outils sont constamment améliorés et l'augmentation des performances des ordinateurs les rendent de plus en plus efficaces. Cependant, nombre de ces outils reposent sur la résolution de problèmes complexes demandant un grand nombre d'opérations. Dans le cas où la variante décisionnelle du problème est NP-complète, ce nombre d'opérations est exponentiel en fonction de la taille du problème. Dans ce cas, la résolution de tels problèmes peut prendre plusieurs centaines d'années.

La mise en place de techniques dédiées permettant de prendre en considération les propriétés intrinsèques du problème est donc un axe de recherche majeur dans le domaine de l'informatique. Parmi les classes de problèmes les plus étudiés, on retrouve les problèmes d'ordonnancement, les problèmes de tournées de véhicules ou les problèmes d'affectation.

Dans cette thèse, nous nous sommes intéressé aux problèmes d'ordonnancement et, plus particulièrement, aux problèmes d'ordonnancement avec contraintes de ressource. Parmi les problèmes les plus étudiés dans la littérature, nous retrouvons le problème d'ordonnancement de projet avec contraintes de ressource et le problème cumulatif.

Dans le problème d'ordonnancement de projet avec contraintes de ressource, nous devons ordonnancer un ensemble d'activités, chacune d'entre elles consommant une partie d'une ou plusieurs ressources (de capacité limitée) et étant liées par des relations de précédence. Le plus souvent, ces activités doivent être ordonnancées de manière à minimiser la date de fin du projet mais de nombreux autres objectifs, tels que la minimisation du coût ou des retards peuvent être rencontrés dans la littérature.

Dans le problème cumulatif, les activités consomment une quantité d'une et une seule ressource. Il s'agit de la même ressource pour toutes les activités. Dans ce problème, il n'y a pas de relation de précédence entre les activités mais chaque activité dispose d'une fenêtre de temps dans laquelle elle doit s'exécuter. Ce problème correspond à une relaxation de la variante décisionnelle du problème d'ordonnancement de projet avec contraintes de ressource.

La plus grande limite des problèmes ci-dessus est que les activités sont supposées de durée fixe et consommant une quantité de ressource constante au cours du temps. Cependant, il existe de nombreuses applications pratiques dans lesquelles ces hypothèses ne sont pas respectées [ALH13, BEP+01, Weg80]. En effet, la possibilité, pour une activité, de consommer plus de ressource (respectivement moins) afin de finir plus rapidement (resp. lentement) n'est pas modélisée dans ces problèmes. Les activités satisfaisant cette propriété sont dites *malléables* dans le sens où leur forme, définie par leur durée et consommation de ressource pendant leur exécution, doit être décidée pendant le processus

de résolution. Des exemples de telles activités sont variés lorsque les activités doivent consommer, par exemple, une ressource de nature énergétique comme l'électricité. La quantité de ressource allouée à une activité peut alors être modulée à tout instant pour accélérer l'activité ou au contraire la ralentir et diminuer sa consommation (souvent pour réduire les coûts énergétiques). Un autre exemple intervient dans l'ordonnancement de projet où la durée d'une activité dépend de la quantité de ressource qui lui est attribuée (p.e. personnel).

Cette thèse étudie une nouvelle modélisation des activités malléables représentée par un problème appelé le problème d'ordonnancement continu avec contraintes énergétiques. Dans ce problème, un ensemble d'activités, utilisant une ressource continue et cumulative de capacité limitée, doit être ordonnancé. La quantité de ressource nécessaire à l'exécution d'une activité n'est pas fixée mais doit être déterminée à chaque instant. Une fois la tâche commencée et jusqu'à sa date de fin, la quantité de ressource consommée par l'activité doit être comprise entre une valeur maximale et une valeur minimale. De telles bornes dans des problèmes pratiques peuvent représenter, par exemple, la quantité d'énergie minimale requise pour qu'une réaction chimique ou thermodynamique puisse se faire dans les conditions prescrites ou encore le nombre minimum et maximum d'employés pouvant être affectés à une activité. De plus, la consommation, à un instant donné, d'une partie de la ressource permet à l'activité de recevoir une certaine quantité d'énergie, calculée par le biais d'une fonction de rendement. Ces fonctions de rendement peuvent, par exemple, représenter les pertes dues à la conversion de la ressource en énergie (p.e. AC/DC) ou les coûts de communication dans des architectures multiprocesseurs. La connaissance de l'énergie reçue par une activité nous permet de savoir quand l'activité est terminée, i.e. quand elle a reçu une quantité suffisante d'énergie.

Pour le problème cumulatif et le problème d'ordonnancement de projet avec contraintes de ressource, différentes techniques permettant de trouver des solutions ont été mises en place dans la littérature. Ces techniques utilisent des concepts et théories pouvant être très variés. Cependant, deux de celles figurant parmi les plus utilisées demeurent les techniques issues de la programmation par contraintes et de la programmation linéaire mixte (ou en nombres entiers). En effet, ces techniques se sont révélées très efficaces dans le processus de résolution de ces deux problèmes. Ce sont quelques-unes de ces méthodes que nous nous proposons d'étendre au problème considéré dans cette thèse.

Le plan de la thèse est le suivant :

— le chapitre 1 commence par détailler les principales caractéristiques des problèmes d'ordonnancement (sous-section 1.1.1). Ensuite, une définition formelle des problèmes d'ordonnancement cumulatif et de projet avec contraintes de ressource ainsi qu'une description des principales limitations de ces problèmes en termes de modélisation de certaines ressources sont données (sous-section 1.1.2). Enfin, la section 1.2 décrit le problème étudié dans ce manuscrit : le problème d'ordonnancement continu avec contraintes énergétiques. De plus, cette section décrit les modélisations préexistantes des activités malléables tout en expliquant en quoi ces modélisations ne suffisaient pas à la modélisation de certains problèmes réels. La dernière partie du chapitre est consacrée à la présentation d'un certain nombre de propriétés remarquables satisfaites par ce problème.

— les chapitres 2 et 3 sont dédiés aux méthodes de programmation par contraintes. Après une brève introduction à la programmation par contraintes (section 2.1) et à l'ordonnancement en programmation par contraintes, nous présentons les principaux algorithmes de filtrage mis en

place pour le problème cumulatif (section 2.2). Dans le chapitre 3, nous adaptons une partie de ces algorithmes au problème d'ordonnancement continu avec contraintes énergétiques. Dans un premier temps, nous montrons qu'une partie de ces algorithmes peut facilement être adaptée en considérant l'ordonnancement des activités dans le pire des cas en termes de durée ou de consommation de ressource (voir section 3.1). La section 3.2 est consacrée à l'adaptation du raisonnement énergétique. Pour ce raisonnement, nous présentons plusieurs méthodes permettant de caractériser les intervalles sur lesquels appliquer ce raisonnement. Nous attirons ici l'attention du lecteur sur l'utilisation du terme *énergie*. En effet, dans ce manuscrit nous utiliserons à la fois ce terme pour le raisonnement énergétique (algorithme de filtrage pour la contrainte cumulative) mais aussi dans un problème qui modélise des ressources énergétiques telles que l'électricité.

— les chapitres 4 et 5 présentent les techniques de résolution issues de la programmation linéaire mixte. Dans un premier temps, nous décrivons les concepts généraux de la programmation linéaire mixte (section 4.1). Nous présentons ensuite trois modèles mis en place pour résoudre le problème d'ordonnancement de projet avec contraintes de ressource (section 4.2). Le premier est un modèle indexé par le temps et les deux autres utilisent des formulations basées sur les événements. Ces modèles sont ensuite adaptés au problème d'ordonnancement continu avec contraintes énergétiques (section 5.1). La section 5.2 présente plusieurs ensembles d'inégalités permettant de renforcer les modèles présentés. Pour le modèle indexé par le temps, des inégalités directement déduites du raisonnement énergétique sont exhibées. Pour les modèles à événements, cinq jeux d'inégalités sont présentés et l'un d'entre eux est utilisé pour donner une description minimale de l'enveloppe convexe du polyèdre formé par toutes les affectations possibles des variables binaires correspondant à une seule activité.

— le chapitre 6 présente les résultats expérimentaux conduits pour valider les notions théoriques décrites dans les chapitres précédents. Dans un premier temps, nous présentons les instances sur lesquelles les algorithmes ont été appliqués (voir section 6.1). La section 6.2 présente les performances des différents modèles de programmation linéaire mixte définis pour le problème d'ordonnancement continu à contrainte énergétique. De plus, l'influence des inégalités définies pour renforcer les modèles est évaluée. La section suivante (6.3) présente les résultats obtenus lors des expérimentations portant sur la programmation par contraintes. Les raisonnements présentés dans le manuscrit sont intégrés dans une méthode de branchement hybride utilisant un modèle de programmation linéaire.

— les annexes A et B présentent des travaux réalisés pendant la thèse mais pas assez aboutis ou trop éloignés du sujet de ce manuscrit pour y figurer à part entière. L'annexe A présente l'étude du cas discret du problème (article publié aux Journées Francophones de Programmation par Contraintes [NAL16]). L'annexe B porte sur la mise en place d'une matheuristique pour un problème industriel d'ordonnancement avec contraintes et objectifs énergétiques. Ce travail s'inscrit dans le cadre d'une collaboration pour un projet franco-chilien et a été publié dans la conférence IESM (International Conference on Industrial Engineering and Systems Management [NAL+15c]).

# Première partie

# Présentation du problème


**1 Ordonnancement, ressources cumulatives et énergie** **19**

1.1 Ordonnancement et contraintes de ressources . . . . . 19

1.1.1 L'ordonnancement . . . . . 19

1.1.2 Contraintes de ressources . . . . . 21

1.2 L'ordonnancement sous contraintes énergétiques . . . . . 26

1.2.1 Définition du problème . . . . . 26

1.2.2 Autres modélisations des activités à profil variable . . . . . 30

1.2.3 Propriétés du CECSP . . . . . 32

*La première partie de cette thèse est consacrée à la présentation de la problématique et de son contexte général. Nous commençons donc, dans un premier temps, par définir les principales caractéristiques d'un problème d'ordonnancement et plus précisément des problèmes d'ordonnancement à contraintes de ressource. Pour ces derniers, deux exemples de tels problèmes sont présentés : le problème d'ordonnancement de projet à contraintes de ressource et une de ses relaxations, le problème cumulatif. En effet, plusieurs techniques de résolution adaptées de celles définies pour ces problèmes seront présentées dans cette thèse.*

*La suite du manuscrit discute des limitations de ces problèmes et présente une nouvelle modélisation permettant de pallier ces dernières : le problème d'ordonnancement continu à contraintes énergétiques [AL15, NALR16, NAL15b]. Ce problème est ensuite comparé aux modélisations préexistantes afin de montrer la pertinence de celui-ci et de souligner que l'application directe des techniques de résolution de ces modélisations à ce problème n'est pas possible.*

*Enfin, la fin de cette partie est dédiée à la présentation des principales propriétés du problème d'ordonnancement continu à contraintes énergétiques. Après avoir prouvé la NP-complétude de ce problème dans le cas général, nous montrons que, même dans le cas où tous les paramètres du problème sont entiers, l'ensemble des solutions peut ne comporter que des valeurs fractionnaires. Nous prouvons ensuite que, dans certains cas particuliers, l'ensemble des instants pour lesquels la ressource doit être réallouée peut être restreint aux dates de début et de fin des activités. Ceci nous permettra, dans un premier temps, de décrire des cas particuliers de ce problème solvable en temps polynomial. Cette propriété sera ensuite utilisée pour définir des méthodes de résolution dans les parties suivantes de cette thèse.*

*Les propriétés définies dans la sous-section 1.2.3 ont été publiées dans [NALR16, NAL15b, NAL15a].*

# Chapitre 1

# Ordonnancement, ressources cumulatives et énergie

Dans ce chapitre, nous commençons par définir ce qu'est un problème d'ordonnancement et quelles sont les principales caractéristiques d'un tel problème. Ensuite, nous présenterons deux exemples de problèmes d'ordonnancement cumulatif que nous utiliserons tout au long de ce manuscrit, le problème d'ordonnancement de projet à contraintes de ressources (RCPSP) et le problème cumulatif (CuSP). Nous montrerons ensuite les limites de ces problèmes en termes de modélisation des modalités d'utilisation de certaines ressources et nous introduirons une nouvelle extension du CuSP, le problème d'ordonnancement continu à contraintes énergétiques (CECSP), pour pallier ces limites. Cette nouvelle extension sera ensuite comparée aux extensions existantes du RCPSP et du CuSP. Enfin, plusieurs propriétés du CECSP, qui seront utilisées dans la suite de ce manuscrit, seront présentées.

## 1.1 Ordonnancement et contraintes de ressources

### 1.1.1 L'ordonnancement

La théorie de l'ordonnancement s'intéresse au calcul de dates d'exécution d'un ensemble d'activités. Dans cette optique, l'utilisation d'une ou plusieurs ressources peut être nécessaire et l'exécution d'une activité implique souvent une telle consommation. Un problème d'ordonnancement peut alors être vu comme l'organisation dans le temps de la réalisation d'activités soumises à des contraintes de temps et de ressource. Dans la plupart des cas, un ou plusieurs objectifs sont définis et une solution au problème d'ordonnancement vise à optimiser ces objectifs.

#### Les activités

Une activité peut être définie par une date de début $st_i$ et une date de fin $et_i$ et une durée $p_i$ vérifiant $et_i = st_i + p_i$. Cette date de début (respectivement fin) doit être comprise entre sa date de début (resp. fin) au plus tôt, $est_i$ (resp. $eet_i$) et sa date de début (resp. fin) au plus tard, $lst_i$ (resp. $let_i$).

Si l'activité utilise une ou plusieurs ressources durant son exécution, il est nécessaire d'ajouter à cette définition une fonction permettant de modéliser l'allocation de chaque ressource $k$ à chaque activité $i$. Si cette fonction est donnée dans les paramètres du problème, on la note $r_{ik}(t)$. Si, au contraire, elle fait partie des variables de décision du problème, on la note $b_{ik}(t)$. Enfin, si cette fonction ne dépend pas du temps, i.e. est constante, le paramètre $t$ pourra être omis.

Selon les problèmes, une activité peut être contrainte à s'exécuter en un seul morceau. On parle alors d'activité non préemptive et dans le cas contraire, i.e. les activités peuvent être exécutées en plusieurs morceaux, on parle d'activité préemptive.

Une activité peut être utilisée pour représenter, par exemple, une opération dans un processus de production, le décollage/atterrissage d'un avion ou encore une étape d'un projet de construction.

**Les ressources**

Une ressource $k$ est un moyen technique ou humain requis pour la réalisation d'une activité et est disponible en quantité limitée. Cette quantité, appelée *disponibilité de la ressource ou capacité*, peut être soit constante ou varier au cours du temps. Dans ce manuscrit, nous considérons des ressources à capacité constante et cette capacité est notée $R_k$. Les ressources utilisées par les activités peuvent être de nature diverse. Parmi elles, on peut distinguer :

— les ressources renouvelables : ces ressources peuvent être réutilisées dès lors qu'elles sont libérées. Il s'agit, en fait, de ressources qui, après avoir été utilisées par une ou plusieurs activités, sont de nouveau disponible en même quantité. Ces ressources peuvent, par exemple, représenter la main d'œuvre d'une entreprise, des machines, de l'électricité ou des équipements.

— à l'inverse, les ressources consommables sont des ressources dont la consommation globale est limitée au cours du temps. Il peut s'agir, par exemple, de matières premières ou d'un budget.

Parmi les ressources renouvelables, on distingue par ailleurs, les ressources disjonctives qui ne peuvent exécuter qu'une activité à la fois – e.g. pistes de décollage, salles – et les ressources cumulatives qui peuvent, elles, être utilisées par plusieurs activités en parallèle mais sont disponibles en quantité limitée – e.g. main d'œuvre, processeurs.

Du point de vue de leur divisibilité, les ressources peuvent aussi être divisées selon deux catégories :

— les ressources continues, i.e. divisibles en temps ou en quantité continu : dans le premier cas, il s'agit de ressources pouvant être ré-allouées à tout instant $t \in [0, T]$, où $T$ est une borne supérieure sur la date de fin de l'ordonnancement ; dans le second cas, il s'agit de ressources pouvant être allouées en quantité continue, i.e. non discrète. Ce type de ressource permet, par exemple, de modéliser l'électricité, l'essence, l'énergie hydraulique...

— les ressources discrètes, i.e. divisibles en temps ou en quantité discret : à l'inverse, le premier cas décrit une ressource où la ré-allocation de cette dernière ne peut être exécutée qu'à des temps discrets $t \in \{0, \ldots, T\}$. Ce cas correspond généralement au cas où l'horizon de temps a été discrétisé, soit pour faciliter le problème, soit sans perte de génralité. Le second cas correspond aux ressources ne pouvant être attribuées aux activités qu'en quantité discrète, e.g. employés, machines...

**Les contraintes**

Une contrainte permet d'exprimer des restrictions sur les valeurs que peuvent prendre une ou plusieurs variables du problème. Parmi les principales, on distingue :

— les contraintes de temps : elles intègrent les contraintes de temps alloué, issues généralement d'impératifs de gestion et relatives aux dates limites des activités (e.g. dates de livraison) ou à la durée totale d'un projet mais aussi les contraintes d'enchaînement qui décrivent des

positionnements relatifs devant être respectés entre les activités. Ces contraintes peuvent, par exemple, modéliser des contraintes de précédence entre les activités, i.e. une activité ne peut commencer avant qu'une autre n'ait été achevée, ou des temps de transition à respecter entre les activités.

Étant donné une activité $i$, la date à partir de laquelle l'activité $i$ peut être exécutée est appelée *date de début au plus tôt* et est notée $est_i$ (earliest start time en anglais). De même, la date avant laquelle l'activité $i$ doit avoir été complètement exécutée sera appelée *date de fin au plus tard* et notée $let_i$ (latest end time en anglais).

— les contraintes de ressources : ce sont des contraintes d'utilisation des ressources qui expriment la nature et la quantité de moyens utilisés par les activités, ainsi que les caractéristiques d'utilisation de ces moyens. Ces contraintes peuvent aussi représenter des contraintes de disponibilité des ressources qui précisent la nature et la quantité de moyens disponibles au cours du temps.

#### Les objectifs

Lors de la résolution d'un problème d'ordonnancement, deux buts différents peuvent être poursuivis. Le premier vise à trouver une solution réalisable pour le problème tandis que le second cherche à trouver une solution réalisable optimisant un ou plusieurs critères ou objectifs.

Ces objectifs peuvent être liés à différents aspects de la solution. On distingue par exemple :

— les objectifs liés au temps : le temps total d'exécution ou le temps moyen d'achèvement d'un ensemble d'activités peuvent être minimisés, mais aussi les retards (maximum, moyen, somme...) par rapport aux dates de fin au plus tard fixées par le problème.

— les objectifs liés aux ressources : la quantité (maximale, moyenne, pondérée...) de ressources nécessaires pour réaliser un ensemble d'activités peut, par exemple, être minimisée.

— les objectifs liés aux coûts de lancement, de production, de transport, de stockage ou liés aux revenus, aux retours d'investissements...

— les objectifs liés à une énergie, un débit...

Deux exemples de problèmes d'ordonnancement sont présentés dans la sous-section suivante : le RCPSP et le CuSP.

### 1.1.2 Contraintes de ressources

Dans ce manuscrit, nous nous intéressons principalement aux problèmes d'ordonnancement cumulatif. Parmi ces derniers, deux des plus étudiés sont le problème d'ordonnancement de projet à contraintes de ressource (RCPSP) et le problème cumulatif (CuSP). Nous allons donc commencer par présenter ces deux problèmes. En effet, beaucoup des travaux décrits dans ce manuscrit s'appuient sur des résultats mis en évidence pour eux.

Dans un second temps, nous montrerons les limites de ces modélisations pour exprimer certains problèmes cumulatifs réels et les modèles alternatifs mis en place dans la littérature pour pallier ces limitations.

**Le problème d'ordonnancement de projet à contraintes de ressources**

Le problème d'ordonnancement de projet à contraintes de ressources (RCPSP) est un problème d'ordonnancement très général, utilisé pour modéliser certains problèmes pratiques. Le but est d'ordonnancer un ensemble d'activités de telle sorte que les capacités des ressources ne soient pas excédées et qu'un certain critière, ou *fonction objectif*, soit minimisé. Parmi les ressources modélisées, on trouve des ressources telles que des machines, des personnes, des salles, de l'argent ou encore de l'énergie. Pour les fonctions objectif, des quantités telles que la durée totale du projet, le retard ou les coûts peuvent être minimisées.

Formellement, le RCPSP est défini de la manière suivante : nous considérons un ensemble d'activités non préemptives $\mathcal{A} = \{1, \ldots, n\}$ à ordonnancer et un ensemble $\mathcal{R} = \{1, \ldots, m\}$ de ressources discrètes, cumulatives et renouvelables. Chacune de ces ressources $k \in \mathcal{R}$ est disponible tout au long du projet en quantité $R_k$ et, durant son exécution, une activité consomme une quantité constante $r_{ik}$ (pouvant être nulle) de cette ressource. Dans ce problème, une activité $i \in \mathcal{A}$ a une durée fixe $p_i$ et des relations de précédence lient les activités entre elles. Ces relations sont modélisées à l'aide d'un graphe $G = (V, E)$, appelé graphe de précédence, dans lequel l'ensemble des arcs $(i, j) \in E$ représente les relations de précédence, i.e. $(i, j) \in E \Leftrightarrow i$ doit être ordonnancée avant $j$ dans toute solution. Dans ce graphe, l'ensemble des sommets, noté $V = \{0, \ldots, n+1\}$, correspond aux $n$ activités auxquelles on ajoute deux activités fictives 0 et $n + 1$ qui représentent respectivement le début et la fin du projet. Ces activités fictives ne consomment pas de ressource et ont une durée d'exécution nulle. De plus, $E$ contient les arcs suivants :

— $(0, i),\ \forall i \in \mathcal{A}$,
— $(i, n + 1),\ \forall i \in \mathcal{A}$.

Pour ce problème, la fonction objectif la plus rencontrée dans la littérature étant la minimisation de la date de fin du projet, i.e. $C_{max}$, nous considérons principalement cet objectif dans la suite de ce manuscrit. Si un objectif différent est considéré, nous le précisons.

L'objectif du problème est donc de déterminer la date de début $st_i$ de chaque activité $i \in \mathcal{A}$ de telle sorte que :

— à chaque instant $t$, la somme, pour chaque activité, des consommations d'une même ressource $k \in \mathcal{R}$ ne doit pas dépasser la capacité $R_k$ de cette dernière, i.e.

$$\forall t \in \mathcal{H}, \forall k \in \mathcal{R}, \sum_{\substack{i \in \mathcal{A} \\ t \in [st_i, st_i + p_i[}} r_{ik} \leq R_k \tag{1.1}$$

avec $\mathcal{H} = \{0, \ldots, T\}$ définissant l'horizon de temps du projet. $T$ est donc une borne supérieure sur la date de fin du projet.

— les contraintes de précédence sont satisfaites, i.e.

$$\forall (i, j) \in E,\ st_i + p_i \leq p_j \tag{1.2}$$

— la date de fin du projet $C_{max} = \max_{i \in \mathcal{A}} st_i + p_i$ soit minimale.

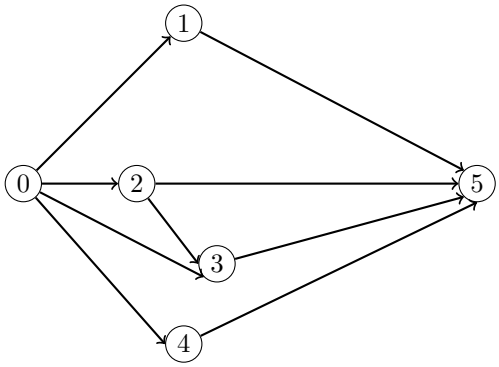
**Exemple 1.1.1.** *Considérons l'instance à quatre activités et deux ressources suivante :*

— $R_1 = 5$ *et* $R_2 = 7$

— *cf. figure 1.1*

| $i$ | $p_i$ | $r_{i1}$ | $r_{i2}$ |
|---|---|---|---|
| 1 | 4 | 2 | 3 |
| 2 | 3 | 1 | 5 |
| 3 | 5 | 2 | 2 |
| 4 | 8 | 2 | 4 |

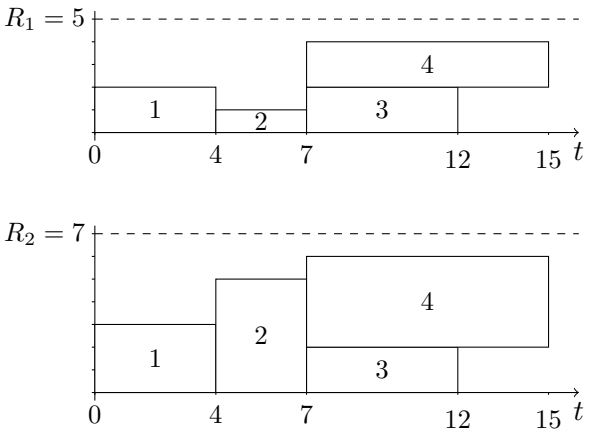
(a) Durée et consommation de chaque activité.



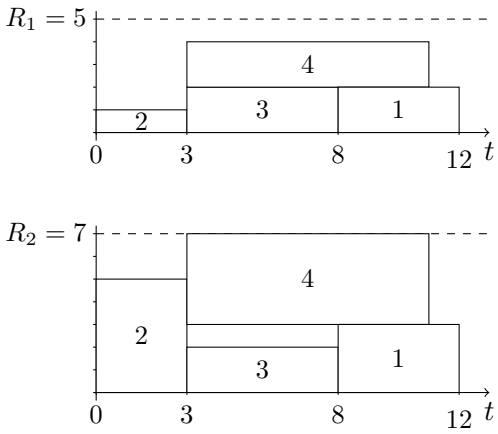

(b) Graphe de précédence des activités.

FIGURE 1.1 – Exemple d'instance pour le RCPSP.

*La figure 1.2a présente un ordonnancement réalisable avec $C_{max} = 15$. Cet ordonnancement n'est pas optimal puisque si l'activité 1 est décalée à droite de manière à commencer au temps $t = 8$, on peut décaler les trois autres activités vers la gauche et on obtient un ordonnancement ayant une date de fin inférieure à celle de l'ordonnancement précédent, i.e. $C_{max} = 12$ (cf. figure 1.2b) dont on peut prouver l'optimalité.*




(a) Une solution réalisable




(b) Une solution optimale

FIGURE 1.2 – Exemple de solutions réalisables pour le RCPSP.

Le RCPSP a été prouvé NP-complet au sens fort [BLK83]. Ce problème a donc été très étudié dans la littérature, notamment pour trouver des méthodes efficaces pour sa résolution. Dans la section 4.2, nous présentons des modèles de programmation linéaire permettant de trouver une solution optimale à ce problème. Ces modèles seront alors adaptés dans le cadre d'un autre problème d'ordonnancement décrit dans le paragraphe 1.2.

### Le problème cumulatif

Le problème d'ordonnancement cumulatif (CuSP) permet de caractériser le fait que le projet implique une ressource (ou un sous-ensemble de ressources) de nature cumulative. Le CuSP peut

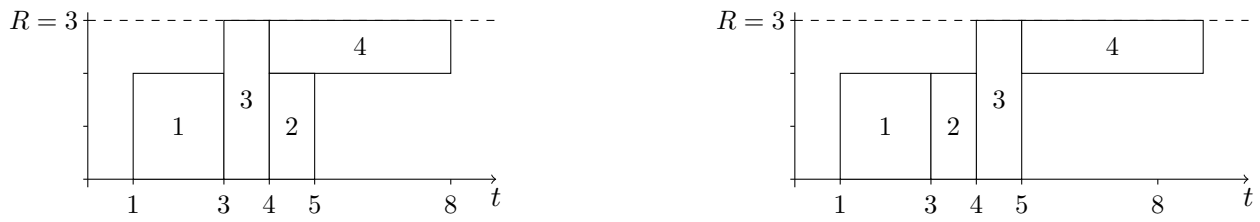



FIGURE 1.3 – Exemple de solutions réalisables pour le CuSP.

être vu comme un cas particulier de la variante décisionnelle du RCPSP où l'on ne considère qu'une ressource et où l'on remplace les contraintes de précédence par les fenêtres de temps qu'elles induisent.

Formellement, le CuSP prend en entrée un ensemble $\mathcal{A} = \{1, \ldots, n\}$ d'activités non préemptives à ordonnancer. Pour s'exécuter, une activité doit consommer une partie de la ressource $r_i$ et ce jusqu'à l'arrêt de l'activité, i.e. après un temps $p_i$ correspondant à la durée de l'activité $i$. Cette ressource est de type cumulatif, discrète et renouvelable, disponible en quantité $R$.

De plus, chaque activité dispose d'une fenêtre de temps $[est_i, let_i]$ dans laquelle l'activité doit obligatoirement s'exécuter. Nous rappelons que $est_i$ correspond à la date de début au plus tôt de $i$ et $let_i$ à sa date de fin au plus tard.

L'objectif du CuSP est donc de déterminer la date de début $st_i$ de chaque activité $i \in \mathcal{A}$ telle que :
— la capacité de la ressource n'est excédée à aucun moment du projet, i.e.

$$\forall t \in \mathcal{H}, \sum_{\substack{i \in \mathcal{A} \\ t \in [st_i, st_i + p_i[}} r_i \leq R \tag{1.3}$$

— la fenêtre de temps de chaque activité est respectée, i.e.

$$\forall i \in \mathcal{A},\ est_i \leq st_i < st_i + p_i \leq let_i \tag{1.4}$$

Trouver une solution réalisable pour ce problème – étant une généralisation de la variante de décision du problème à une machine ($R = 1$ et $r_i = 1$) et du problème à $m$ machines ($R = m$ et $r_i = 1$) – est NP-complet au sens fort [GJ79]. De ce fait, dans la littérature, ce problème est souvent étudié sans fonction objectif. Sauf indication contraire, nous ferons de même dans la suite du manuscrit.

**Exemple 1.1.2.** *Considérons l'instance à quatre activités suivante :*
— $R = 4$
— *cf. table 1.1*

| $i$ | $p_i$ | $est_i$ | $let_i$ | $r_i$ |
|---|---|---|---|---|
| 1 | 2 | 1 | 5 | 2 |
| 2 | 1 | 3 | 5 | 2 |
| 3 | 1 | 3 | 5 | 3 |
| 4 | 4 | 1 | 10 | 1 |

TABLE 1.1 – Exemple d'une instance du CuSP.

*La figure 1.3 présente plusieurs solutions réalisables pour cette instance.*

Dans la section 2.2, nous présenterons des méthodes de résolution pour ce problème utilisant la programmation par contraintes.

#### Limites des problèmes cumulatifs en termes de modélisation

Une des principales limitations du RCPSP et donc du CuSP (qui en est un cas particulier) en termes de modélisation est que, dans ces problèmes, chaque activité consomme une quantité de ressource fixe, connue à l'avance, durant toute sa durée d'exécution. De plus, cette durée est aussi supposée fixe. Cependant, pour de nombreux problèmes pratiques, ces suppositions reviennent à sur-contraindre le problème. En effet, considérons l'exemple suivant, décrit dans [FT10] pour lequel une activité représentant la peinture d'un bateau est considérée. Pour cette activité, la durée est remplacée par une quantité de travail, ou énergie, représentant le travail de 3 personnes sur une journée. Cette activité peut, au choix, être exécutée en 3 jours par 1 personne, ou en 2 jours par 2 personnes le premier jour et 1 personne le second, ou encore par 3 personnes en seulement un jour. Si l'on avait supposé une consommation et une durée fixe, l'espace des solutions aurait été réduit.

L'exemple décrit ci-dessus peut facilement être généralisé à de nombreux cas pratiques. Un autre exemple venant d'un problème industriel est présenté dans [ALH13]. Dans cet article, une application dans une fonderie où du métal est fondu dans des fours à induction est détaillé. La puissance de chaque four utilisée pour faire fondre le métal peut être réglée, à tout moment, afin d'éviter un dépassement d'un certain niveau de ressource (généralement dû à une limite suggérée par le fournisseur). De ce fait, si chaque opération de fonte est vue comme une activité, nous avons besoin de pouvoir moduler la quantité de ressource donnée à cette activité à chaque instant et la durée de l'activité dépend de cette quantité de ressource.

Pour représenter ces variations dans le profil de consommation des activités, nous avons choisi de modéliser la ressource comme une ressource continue. En effet, de nombreux exemples de ressource continue existent. C'est le cas de l'électricité, des sources d'énergie hydrauliques, du carburant ou encore de la mémoire d'un ordinateur. D'autres ressources telles que des employés ou des machines, qui sont normalement modélisées à l'aide de ressources discrètes, peuvent être modélisées par des ressources continues si l'on suppose qu'un employé ou une machine peut exécuter plusieurs activités en parallèle [Weg80, NK14].

De plus, les problèmes à ressources continues peuvent aussi servir de relaxation pour les problèmes avec des ressources discrètes. En effet, le caractère discret du problème peut parfois amener à considérer de nombreuses possibilités d'affectations de la ressource. Dans certains cas, ce grand nombre d'affectation peut grandement complexifier le problème. Donc, considérer des ressources continues peut permettre l'agrégation des raisonnements dédiés à ces problèmes.

Une autre particularité du problème de la fonderie décrit dans [ALH13] vient du fait que la quantité de ressource allouée à un four doit être comprise dans un certain intervalle pendant toute la durée d'exécution de l'activité. En effet, pour des raisons opérationnelles et/ou physiques, la puissance des fours permettant la fonte du métal ne peut être ni trop élevée, ni trop basse et, de ce fait, ne peut ni dépasser un certain seuil dit maximum, ni descendre en dessous d'un seuil minimum.

Enfin, la quantité d'énergie fournie peut ne pas être proportionnelle à la quantité de ressource

consommée. Le problème considéré dans cette thèse et décrit dans la section suivante permet de modéliser ces contraintes particulières. Cette modélisation sera ensuite comparée aux extensions du CuSP et du RCPSP, déjà existantes dans la littérature.

## 1.2 L'ordonnancement sous contraintes énergétiques

La modélisation présentée dans cette section repose sur le problème d'ordonnancement continu à contraintes énergétiques [AL15], le CECSP.

### 1.2.1 Définition du problème

Dans ce problème, un ensemble d'activités non préemptives $\mathcal{A} = \{1, \ldots, n\}$ utilisant une ressource continue, cumulative et renouvelable, de capacité $R$ doit être ordonnancé. Durant son exécution, une activité consomme une quantité variable $b_i(t)$ de la ressource qui doit être comprise entre une valeur minimale, $r_i^{min} \in [0, R]$, et une valeur maximale, $r_i^{max} \in [r_i^{min}, R]$. De plus, la fin d'une activité correspond au moment où cette dernière a reçu une certaine quantité d'énergie $W_i$. Cette énergie est reçue via la ressource et calculée à l'aide d'une fonction $f_i : \{0\} \cup [r_i^{min}, r_i^{max}] \longrightarrow \{0\} \cup [f(r_i^{min}), f(r_i^{max})]$ ($f_i(0) = 0$ est imposé). Ces fonctions, appelées fonctions de rendement, font partie de la donnée du problème et sont supposées continues et strictement croissantes. La quantité d'énergie reçue par $i$ à l'instant $t$ est donc $\int_0^t f_i(b_i(s))ds$. La dernière contrainte du problème précise que chaque activité doit être exécutée dans sa fenêtre de temps $[est_i, let_i]$.

L'objectif du CECSP est donc de déterminer la date de début $st_i$ et de fin $et_i$ de chaque activité $i \in \mathcal{A}$, ainsi que la fonction d'allocation de ressource, $b_i(t)$, associée à cette activité telle que :

— la fenêtre de temps de chaque activité est respectée, i.e.

$$\forall i \in \mathcal{A},\ est_i \leq st_i < et_i \leq let_i \tag{1.5}$$

Les activités de durée nulle ne sont pas considérées.

— la capacité de la ressource n'est excédée à aucun moment du projet, i.e.

$$\forall t \in \mathcal{H},\ \sum_{\substack{i \in \mathcal{A} \\ t \in [st_i, et_i[}} b_i(t) \leq R \tag{1.6}$$

où $\mathcal{H} = \{0, \ldots, T\}$ est l'horizon de temps du projet et $T = \max_{i \in \mathcal{A}} let_i$.

— si une activité est en cours à l'instant $t$ alors les contraintes de consommation minimale et maximale doivent être respectées, i.e.

$$\forall i \in \mathcal{A},\ \forall t \in [st_i, et_i[,\ r_i^{min} \leq b_i(t) \leq r_i^{max} \tag{1.7}$$

— si l'activité n'est pas en cours, alors elle ne consomme pas de ressource, i.e.

$$\forall i \in \mathcal{A},\ \forall t \notin [st_i, et_i[,\ b_i(t) = 0 \tag{1.8}$$

— l'énergie requise doit être apportée à chaque activité, i.e.

$$\forall i \in \mathcal{A},\ \int_{st_i}^{et_i} f_i(b_i(t))dt = W_i \tag{1.9}$$

Dans certains cas, nous pourrons remplacer cette contrainte par la contrainte suivante :

$$\forall i \in \mathcal{A},\ \int_{st_i}^{et_i} f_i(b_i(t))dt \geq W_i \tag{A.4a}$$

C'est par exemple le cas quand $b_i(t)$ ou $st_i$ et $et_i$ sont contraints à prendre des valeurs entières.
Dans ce manuscrit, nous considérons les cas où les fonctions $f_i$ sont continues, croissantes et :
— égales à la fonction identité, $\forall i \in \mathcal{A}$,
— affines, $\forall i \in \mathcal{A}$,
— concaves et affines par morceaux, $\forall i \in \mathcal{A}$.

L'intérêt de considérer de telles fonctions de rendement est double. Premièrement, un certain nombre de fonctions de rendement réelles ont une forme concave [MHM05, Lew08] due au fait, qu'à partir d'un certain seuil, il n'est plus aussi "rentable" d'allouer plus de ressource à une activité. Deuxièmement, les fonctions affines et concaves affines par morceaux nous permettent d'approcher un grand nombre de fonctions de rendement réelles non linéaires. Un exemple présentant de telles approximations sera présenté (cf. exemple 1.2.1).

Soit $P_i$ le nombre d'intervalles de définition de la fonction $f_i$, i.e. le nombre d'intervalles où la fonction $f_i$ a une expression différente, et $\mathcal{P}_i = \{1, \ldots, P_i\}$ l'ensemble des indices de ces intervalles. Les points de cassures de la fonctions $f_i$ sont notés $x^i_\ell$, $\forall \ell \in \mathcal{P}_i$. La fonction $f_i$ peut alors s'écrire de la manière suivante :

$$f_i(b) = \begin{cases} 0 & \text{si } b = 0 \\ a_{i1} * b + c_{i1} & \text{si } r_i^{min} = 0 \text{ et } b \in ]r_i^{min}, x_2^i] \\ a_{i1} * b + c_{i1} & \text{si } r_i^{min} \neq 0 \text{ et } b \in [r_i^{min}, x_2^i] \\ a_{i\ell} * b + c_{i\ell} & \text{si } b \in ]x_\ell^i, x_{\ell+1}^i], \ell \in \mathcal{P}_i \setminus \{1\} \end{cases}$$

De plus, nous considérons que la fonction $f_i$ satisfait les propriétés suivantes :
— $a_{i1} > a_{i2} > \cdots > a_{iP_i} > 0$ et $c_{i1} < c_{i2} < \cdots < c_{iP_i}$ pour assurer la croissance et la concavité de la fonction ;
— $-a_{i1} * r_i^{min} \geq c_{i1}$ afin de s'assurer que $f_i(b) \geq 0,\ \forall b \in [r_i^{min}, r_i^{max}]$ ;
— $a_{i\ell} * b + c_{i\ell} = a_{i\ell+1} * b + c_{i\ell+1}$ pour assurer la continuité de la fonction.

Dans le cas où $f_i(b)$ est une fonction affine, on pourra noter : $f_i$ :

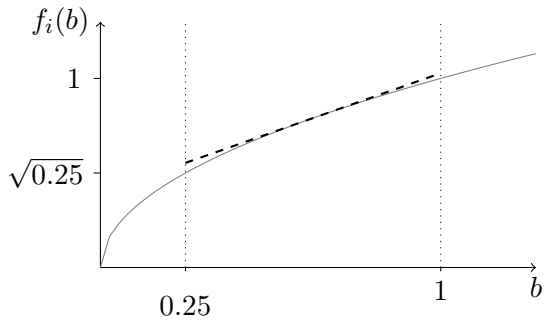
$$f_i(b) = \begin{cases} 0 & \text{if } b = 0 \\ a_i * b + c_i & \text{if } r_i^{min} = 0 \text{ and } b \in ]r_i^{min}, r_i^{max}] \\ a_i * b + c_i & \text{if } r_i^{min} \neq 0 \text{ and } b \in [r_i^{min}, r_i^{max}] \end{cases}$$

**Exemple 1.2.1.** *Considérons l'instance à quatre activités suivante :*
— $B = 2$
— $est = (0, 2, 0, 5)$
— $let = (6, 10, 9, 13)$
— $r^{min} = (0, 0.25, 2, 1)$

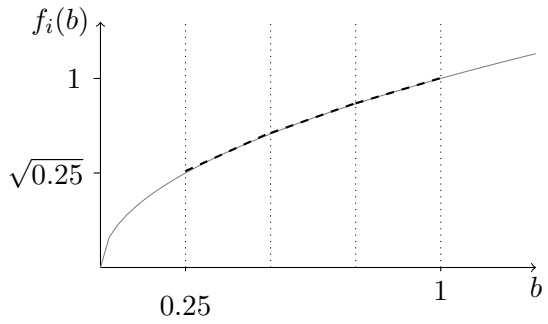
— $r^{max} = (1, 1, 2, 1.5)$
— $W = (1, 5, 7, 8)$
— $f(b) = (b, \sqrt{b}, b, \sqrt{b})$

*Le but de cet exemple étant d'illustrer l'approximation d'une fonction non affine ou concave affine par morceaux par une fonction vérifiant cette propriété, aucune fonction objectif n'est définie pour cette instance.*

*Nous devons approcher les fonctions* $f_2(b)$ *et* $f_4(b)$*. Commençons par approcher l'activité* 2 *par une fonction affine. Pour cela, nous calculons le coefficient directeur de la tangente en* $r_i^{min} + (r_i^{max} - r_i^{min})/2 = 0.625$*. Ce coefficient directeur est égal à* $\frac{1}{2\sqrt{0.625}}$*, et donc,* $f_2'(b) = \frac{1}{2\sqrt{0.625}} * b + \frac{\sqrt{0.625}}{2}$ *(cf. figure 1.4a).*




(a) Approximation par une fonction affine




(b) Approximation par une fonction concave, affine par morceaux

FIGURE 1.4 – Exemple d'approximation d'une fonction de rendement non linéaire par une fonction affine et par une fonction concave et affine par morceaux.

*De même, pour l'activité* 4*, nous avons* $f_4'(b) = \frac{1}{2\sqrt{1.25}} * b + \frac{\sqrt{1.25}}{2}$.

*Approchons maintenant la fonction* $f_2(b)$ *par une fonction concave et affine par morceaux. Dans un premier temps, nous devons choisir le pas d'approximation* $\epsilon$*, i.e. la taille des intervalles pour lesquels la fonction* $f_i$ *a une expression différente. Dans cet exemple, nous choisissons* $\epsilon = 1/4$*. Le nombre d'intervalles de définition de la fonction* $f_2$ *est alors* $(r_i^{max} - r_i^{min})/\epsilon = 3$*. Pour chacun de ces intervalles, nous appliquons la procédure utilisée pour l'approximation de* $f_2$ *par une fonction affine. Nous obtenons donc l'approximation suivante (cf. figure 1.4b) :*

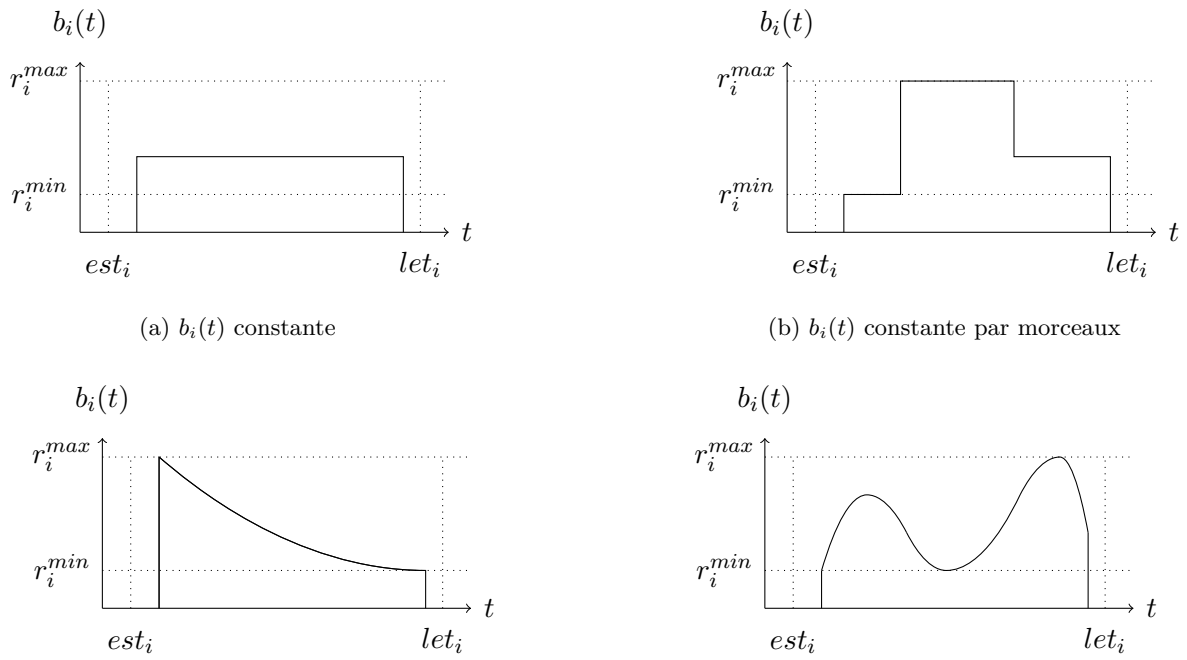
$$f_2 = \begin{cases} \frac{1}{2\sqrt{3/8}} * b + \frac{\sqrt{3/8}}{2} & si\ b \in [0.25, 0.5] \\ \frac{1}{2\sqrt{5/8}} * b + \frac{\sqrt{5/8}}{2} & si\ b \in [0.5, 0.75] \\ \frac{1}{2\sqrt{7/8}} * b + \frac{\sqrt{7/8}}{2} & si\ b \in [0.75, 1] \end{cases}$$

Avec une telle approche, il peut arriver que la fonction de rendement $f_i$ ne vérifie pas $r_i^{min} = 0 \Rightarrow f_i(r_i^{min}) = 0$. Dans ce cas, la valeur de $f_i(0)$ est mise à 0. La fonction n'est donc plus continue sur tout son intervalle de définition. La contrainte (A.4) est donc remplacée par :

$$\int_{st_i}^{et_i} \mathbf{1}_{NZ}(t) f_i(b_i(t)) dt = W_i \qquad \text{(A.4b)}$$

où $\mathbf{1}_{NZ}(t) := \begin{cases} 1 & \text{si } t \in NZ := \{t | b_i(t) \neq 0\} \\ 0 & \text{sinon} \end{cases}$ est la fonction caractéristique de l'ensemble $\mathbb{R}^+$.

La difficulté du CECSP repose, entre autres choses, sur le fait que la fonction d'allocation de ressource $b_i(t)$ peut n'être ni constante, ni constante par morceaux. De ce fait, la représentation temps/ressource d'une activité peut prendre n'importe quelle forme (cf. figure 1.5).




(a) $b_i(t)$ constante

(b) $b_i(t)$ constante par morceaux

(c) $b_i(t)$ quelconque

FIGURE 1.5 – Différentes formes de fonction d'allocation de ressource pour le CECSP.

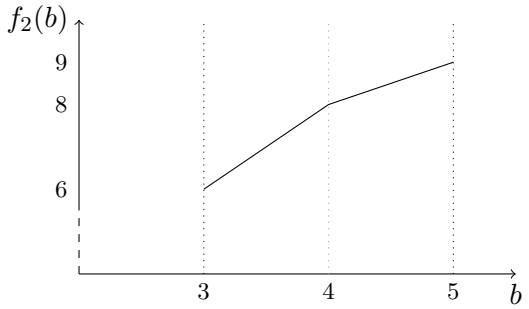
Nous allons maintenant décrire un exemple d'instance et de solution pour le CECSP. Cependant, par souci de clarté, nous présentons un exemple où il existe une solution dans laquelle toutes les fonctions $b_i(t)$ sont constantes par morceaux.

**Exemple 1.2.2.** *Considérons l'instance à trois activités du CECSP suivante :*

- $R = 5$
- *cf. figure 1.6*
- *la fonction* $f_2(b)$ *est définie par l'expression suivante :*

$$f_2(b) = \begin{cases} 2b & b \in [3, 4] \\ b + 4 & b \in ]4, 5] \end{cases}$$

*La figure 1.7 présente une solution réalisable pour le CECSP. Dans cette figure, nous pouvons voir que l'énergie reçue par une activité n'est, a priori, pas égale à la quantité de ressource consommée par cette dernière. En effet, regardons l'activité* 2*. Sa consommation de ressource sur l'intervalle* $[2, 3[$ *est de* 3*. Sur cet intervalle, l'énergie reçue par l'activité est alors de* $f_2(3) = 6$*. Sur les intervalles* $[3, 4[$ *et* $[4, 5[$*, l'activité consomme* 4 *unités de ressource et reçoit une énergie de* $f_2(4) = 8$*. Au total, l'activité* 2 *consomme* $3 + 4 + 4 = 11$ *unités de ressource et reçoit une quantité d'énergie égale à* $f_2(3) + f_2(4) + f_2(4) = 6 + 8 + 8 = 22$*.*




(a) Fonction $f_2(b)$

| $i$ | $r_i$ | $d_i$ | $W_i$ | $r_i^{min}$ | $r_i^{max}$ | $f_i(b)$ |
|---|---|---|---|---|---|---|
| 1 | 0 | 2 | 6 | 3 | 3 | $b$ |
| 2 | 1 | 5 | 22 | 3 | 4 | fig. 1.6a |
| 3 | 0 | 6 | 39 | 1 | 5 | $3b$ |

(b) Données de l'instance

FIGURE 1.6 – Exemple d'une instance pour le CECSP.

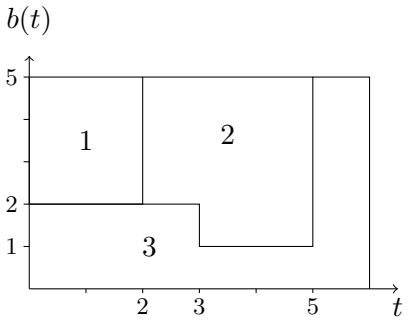



FIGURE 1.7 – Exemple de solution pour le CECSP.

La sous-section suivante présente les différentes modélisations des activités à profil variable présentes dans la littérature.

### 1.2.2 Autres modélisations des activités à profil variable

Dans un premier temps, nous nous intéressons aux extensions du RCPSP. Une des extensions les plus célèbres est le problème d'ordonnancement de projet multimode (MRCPSP). Dans ce problème, un choix de différents modes est disponible pour chaque activité et une activité doit être exécutée selon un de ces modes. Un mode correspond à une combinaison formée d'un temps d'exécution constant et d'une consommation de ressource qui permet d'apporter à l'activité au moins la quantité d'énergie requise. Même si de nombreux problèmes basés sur ce concept de mode existent [DRDH98, RK07, RRK09, DDrH00] et que des méthodes de résolution efficaces ont été mises en place pour résoudre le MRCPSP [PV10], cette modélisation peut amener à une mauvaise allocation de la ressource.

Si nous reprenons l'exemple de la peinture d'un bateau, décrit à la sous-section 1.1.2, l'activité avait besoin de 3 unités d'énergie pour s'exécuter. Dans le contexte du MRCPSP, seulement 3 modes seraient décrits : $(3, 1)$, $(2, 2)$ et $(1, 3)$. Or, dans le second cas, on donne une unité de trop à l'activité et la possibilité d'allouer 2 unités de ressource pendant une période de temps et 1 unité pendant la seconde n'est pas représentée ici.

La principale limitation du MRCPSP est donc que les activités sont contraintes à être rectangulaires, i.e. avec une consommation de ressource constante.

D'autres extensions du RCPSP existent. C'est le cas par exemple des problèmes d'ordonnancement

de projet avec une ressource de type *work-content* [FT10] ou du problème d'ordonnancement de projet avec des profils de ressource flexibles (FRCPSP) [NK14]. Dans ces problèmes, plusieurs types de ressources sont considérées :

- principale (ou work-content dans [FT10]) : il s'agit de la ressource via laquelle la quantité d'énergie requise est donnée à l'activité. C'est elle qui sert à déterminer la durée de l'activité.
- les ressources dépendantes : l'utilisation de ces ressources dépendent de l'utilisation de la ressource principale.
- les ressources indépendantes : la consommation de ces ressources est indépendante des consommations des autres ressources mais ces utilisations doivent être synchrones.

Bien que plusieurs différences existent entre ces problèmes et le CECSP– l'utilisation de plusieurs ressources, ressource/temps discret pour [FT10]... – les principales sont les suivantes : la longueur minimale des blocs et les fonctions de rendement. La première correspond au temps minimal qu'il faut attendre entre deux ré-allocations de la ressource, que les auteurs de [FT10] appellent longueur minimale de bloc et qui est absente dans notre problème. La seconde fait référence à l'absence de fonctions de rendement dans [FT10].

Enfin, la dernière extension du RCPSP présentée est celle où les activités ont une intensité variable [Kis05]. Ici, chaque activité requiert une certaine quantité d'énergie durant toute son exécution. Pour apporter cette énergie à l'activité, il faut décider, dans chaque période de temps, l'*intensité* à laquelle est exécutée l'activité. L'énergie apportée à l'activité est alors proportionnelle à cette intensité. Dans ce cas, on peut introduire des fonctions de rendement mais ces fonctions seraient alors contraintes à être linéaires, i.e. $b \rightarrow a * b$. De plus, aucune borne inférieure sur la consommation d'une activité n'est considérée.

Dans le cadre du CuSP, d'autre variantes ainsi que des algorithmes de filtrages dédiés ont été proposés. Parmi ceux-ci, on retrouve le cas des activités complètement/partiellement élastiques de Baptiste et al. [BLPN99]. Dans le premier cas, les activités ont une demande en énergie constante mais la quantité de ressource consommée par une activité à chaque instant (discret) peut varier entre 0 et la capacité de la ressource. Dans le second cas, les mêmes conditions sont présentes mais les auteurs définissent des contraintes permettant de limiter les variations dans l'utilisation de la ressource. Aucun de ces deux problèmes ne considère de fonctions de rendement.

Dans [BP07], les auteurs définissent une activité comme une séquence de sous-activités trapézoïdales ayant des durées et hauteurs (consommations) variables. Enfin, Vilím [Vil09b] considère des activités pour lesquelles la durée et la hauteur sont définies par des intervalles. Pour ces deux problèmes, aucune demande en énergie n'est définie pour les activités. De plus, dans le second, l'énergie manquante peut être achetée moyennant un certain coût.

Enfin, le CECSP  est aussi lié à d'autres problèmes à contraintes d'énergie avec ressources continues [BEP+01, Wal11]. Dans [BEP+01], plusieurs modèles représentant le temps d'exécution d'une activité en fonction de la ressource qui lui est allouée sont présentés. En particulier, les auteurs considèrent un problème où un ensemble de processeurs identiques et parallèles jouent le rôle de la ressource. De plus, des fonctions représentant le temps d'exécution d'une activité en fonction du nombre de processeurs qui lui est allouée sont définies. Ce nombre de processeurs peut varier continuellement au cours du temps et donc ces fonctions sont équivalentes aux fonctions de rendement définies dans le

cadre du CECSP. De plus, dans [BEP+01, Wal11], l'énergie est calculée en intégrant une fonction de rendement sur tout l'horizon de temps. Cependant, aucune contrainte de consommation maximum et minimum n'est considérée dans ces problèmes. Dans [Wal11], une partie des ressources est continue et une partie est discrète.

Le tableau 1.2 récapitule les principales différences entre tous ces problèmes et le CECSP.

| Problème | $r_i^{min}$ | $r_i^{max}$ | fonction de rend. ($f_i$) | activités non rect. | énergie ($W_i$) | res. cont. | autre différence |
|---|---|---|---|---|---|---|---|
| MRCPSP [DRDH98] | √ | √ | √ | | √ | | |
| FRCPSP [NK14] | √ | √ | | √ | √ | √ | long. de bloc |
| Work-content [FT10] | √ | √ | | √ | √ | | long. de bloc |
| Intensité variable [Kis05] | | √ | | √ | √ | | |
| Partiellement élastique [BLPN01] | | | | √ | √ | | |
| Complètement élastique [BLPN01] | | | √ | √ | | | |
| Activités trapézoïdales [BP07] | | | | √ | | | #trapèzes fixe |
| Représentation par intervalles [Vil09b] | √ | √ | | | | | achat d'énergie |
| Modèle processeurs [BEP+01] | | | √ | √ | √ | √ | |
| Continu/discret [Wal11] | | | √ | √ | √ | √ | res. discrètes et continues |

TABLE 1.2 – Principales différences entre les extensions des problèmes cumulatifs et le CECSP.

Le CECSP est donc un nouveau problème et les différences avec les problèmes existants ne nous permettent pas d'appliquer directement des techniques déjà définies pour d'autres problèmes. Cependant, certaines techniques existantes peuvent être adaptées dans le cadre du CECSP. Ces techniques seront présentées plus tard dans le manuscrit.

La section suivante présente des propriétés du CECSP qui seront utilisées dans le cadre de sa résolution.

### 1.2.3 Propriétés du CECSP

Dans cette section, nous allons commencer par présenter la preuve de NP-complétude du CECSP. Ce problème pouvant être vu comme une généralisation du CuSP, nous utilisons ce problème pour montrer la difficulté du CECSP.

**Théorème 1.1** ([NAL15b]). *Le CECSP est NP-complet.*

*Démonstration.* Nous réduisons donc le CuSP vers le CECSP. Soit $\Pi$ une instance du CuSP. Nous réduisons $\Pi$ en une instance du CECSP, $\Pi'$, de la manière suivante, $\forall i \in \mathcal{A}$ :

— $r_i^{min} = r_i^{max} = r_i$
— $f_i(b) = b$

— $W_i = p_i r_i$
— $R$, $est_i$ and $let_i$ restent inchangés.

On peut facilement vérifier que $\Pi$ est une instance positive du CuSP si et seulement si $\Pi'$ est une instance positive du CECSP. Le CECSP est donc NP-complet. □

Le problème de décision associé au CECSP est donc NP-complet. Dans ce manuscrit, nous avons donc considérer ce problème sans fonction objectif mais aussi avec la fonction objectif suivante :

$$\text{minimiser} \sum_{i \in \mathcal{A}} \int_{st_i}^{et_i} b_i(t)dt$$

Cette fonction consiste en la minimisation de la consommation totale de ressource. L'intérêt de cette fonction objectif dans le cas où les fonctions de rendement sont égales à la fonction identité est discuté dans le chapitre 5. Dans le cas où les fonctions de rendement sont affines ou concaves et affines par morceaux, cet objectif est pertinent puisque, même si la quantité d'énergie apportée à une activité est fixée, la quantité de ressource que l'activité doit consommer ne l'est pas et plusieurs profil de consommation peuvent conduire à la même quantité d'énergie apportée. Trouver le profil qui consomme le moins de ressource possible tout en apportant l'énergie requise est donc un vrai problème. Dans la suite, si rien n'est précisé, cela veut dire que nous considérons le CECSP sans fonction objectif.

Nous présentons un exemple d'instance ne comprenant que des données entières et ne possédant que des solutions non entières. Ceci permet de justifier l'utilisation de modèles à temps continu.

**Exemple 1.2.3.** *Dans cet exemple, nous considérons une instance à deux activités et une ressource de capacité* 2. *Le tableau ci-dessous décrit les données de l'instance :*

| $i$ | $est_i$ | $let_i$ | $W_i$ | $r_i^{min}$ | $r_i^{max}$ | $f_i(b_i(t))$ |
|---|---|---|---|---|---|---|
| 1 | 0 | 2 | 18 | 2 | 2 | $3b_i(t) + 6$ |
| 2 | 1 | 3 | 3 | 1 | 2 | $b_i(t)$ |

*L'unique solution est décrite par la figure 1.8.*

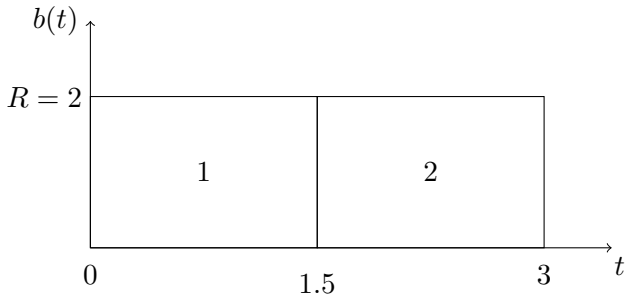



FIGURE 1.8 – Exemple de solution non-entière pour une instance du CECSP à données entières.

*Dans cette solution, la première activité doit finir au temps* $t = 1.5$ *pour que la seconde activité puisse finir avant sa date échue* $let_2 = 3$. *En effet, l'activité* 2 *doit commencer avant sa date de début au plus tard, ici* $lst_2 = let_2 - W_i / f_i(r_i^{max}) = 3 - 3/2 = 1.5$, *et l'activité* 1 *ne peut finir avant sa date de*

*fin au plus tôt, eet*$_1 = est_1 + W_i/f_i(r_i^{max}) = 0 + 18/12 = 1.5$. *De plus, l'activité* 1 *consomme forcément* 2 *unités de ressource durant son exécution* ($r_i^{min} = r_i^{max} = 2$). *Pour apporter exactement l'énergie requise à l'activité* 1, *il faut obligatoirement l'ordonnancer comme sur la figure 1.8.*

De ce fait, l'espace des solutions peut être réduit par l'utilisation du modèle à temps discret mais ceci peut conduire à des infaisabilités ou à des résultats sous-optimaux.

Une solution pour pallier ce problème est de mettre à l'échelle les instances, i.e. multiplier les données par un certain coefficient $\alpha$ afin de s'assurer de l'existence d'une solution optimale entière, avant de les résoudre. Cependant, l'utilisation d'un coefficient trop grand peut conduire à une augmentation de la taille des modèles trop importante pour permettre leur résolution.

Le théorème suivant présente une des propriétés majeures du CECSP. En effet, il stipule que quelle que soit l'instance considérée, il existe toujours une solution de cette instance où les fonctions $b_i(t)$ sont constantes par morceaux, sous certaines conditions sur les fonctions de rendement.

**Théorème 1.2** ([NAL15a]). *Soit* $\Pi$ *une instance réalisable du CECSP telle que :* $\forall i \in \mathcal{A},\ f_i(0) = 0$ *et* $f_i$ *est croissante, continue, concave et affine par morceaux. Une solution ayant la propriété que,* $\forall i \in \mathcal{A},\ b_i(t)$ *soit constante par morceaux, existe.*

Afin de prouver le théorème 1.2, nous commençons par prouver l'affirmation suivante. Soit un intervalle $[t_1, t_2]$ et une fonction d'allocation de ressource $b_i(t)$ non constante dans cet intervalle, alors il existe une constante $b_{iq}$ pour laquelle exécuter $i$ à $b_{iq}$ dans l'intervalle $[t_1, t_2]$, i.e. $b'_i(t) = b_{iq},\ \forall t \in [t_1, t_2]$, apporte au moins autant d'énergie tout en consommant la même quantité de ressource qu'exécuter $i$ à $b_i(t)$ durant l'intervalle $[t_1, t_2]$. C'est ce qu'affirme le lemme suivant :

**Lemme 1.1.** *Soit* $b_{iq} = \frac{\int_{t_1}^{t_2} b_i(t)dt}{t_2 - t_1}$. *Alors, nous avons :*

$$\int_{t_1}^{t_2} b_{iq}dt = \int_{t_1}^{t_2} b_i(t)dt \tag{1.10}$$

$$\int_{t_1}^{t_2} f_i(b_{iq})dt \geq \int_{t_1}^{t_2} f_i(b_i(t))dt \tag{1.11}$$

*Démonstration.* L'équation (1.10) est trivialement vérifiée en remplaçant $b_{iq}$ par sa valeur. En effet, nous avons :

$$\begin{aligned}\int_{t_1}^{t_2} b_{iq}dt &= \int_{t_1}^{t_2} \left(\frac{\int_{t_1}^{t_2} b_i(t)dt}{t_2 - t_1}\right) dt \\ &= (t_2 - t_1)\left(\frac{\int_{t_1}^{t_2} b_i(t)dt}{t_2 - t_1}\right) \\ &= \int_{t_1}^{t_2} b_i(t)dt\end{aligned}$$

Pour prouver que l'équation (1.11) est satisfaite, nous utilisons le théorème suivant :

**Théorème 1.3** ([Jen06]). *Soit* $\alpha(t)$ *et* $g(t)$ *deux fonctions intégrables sur* $[t_1, t_2] \subseteq \mathbb{R}$ *telles que* $\alpha(t) \geq$

0, $\forall t \in [t_1, t_2]$. *Alors, nous avons la propriété suivante :*

$$\phi\left(\frac{\int_{t_1}^{t_2} \alpha(t)g(t)dt}{\int_{t_1}^{t_2} \alpha(t)dt}\right) \geq \frac{\int_{t_1}^{t_2} \alpha(t)\phi(g(t))dt}{\int_{t_1}^{t_2} \alpha(t)dt} \tag{1.12}$$

*où $\phi$ est une fonction continue, concave sur* $[\min_{t\in[t_1,t_2]} g(t), \max_{t\in[t_1,t_2]} g(t)]$.

Si nous remplaçons $\phi(t)$ par $f_i(t)$, $g(t)$ par $b_i(t)$ et $\alpha(t)$ par la fonction constante égale à 1, nous obtenons :

$$\begin{aligned}
&f_i\left(\frac{\int_{t_1}^{t_2} b_i(t)dt}{t_2 - t_1}\right) \geq \frac{\int_{t_1}^{t_2} f_i(b_i(t))dt}{t_2 - t_1} \\
&\Leftrightarrow (t_2 - t_1) f_i\left(b_{iq}\right) \geq \int_{t_1}^{t_2} f_i(b_i(t))dt \\
&\Leftrightarrow \int_{t_1}^{t_2} f_i\left(b_{iq}\right) dt \geq \int_{t_1}^{t_2} f_i(b_i(t))dt
\end{aligned}$$

Et donc, l'équation (1.11) est satisfaite. □

Nous pouvons maintenant prouver le théorème 1.2. Pour cela, nous allons montrer que, soit $S$ une solution d'une instance $\Pi$, alors nous pouvons transformer $S$ en une solution $S'$ ayant la propriété que chaque fonction $b'_i(t)$ est constante par morceaux.

*Preuve du théorème 1.2.* Soit $S$ une solution réalisable de $\Pi$ et soit $(t_q)_{q=1..Q}$ la suite des différentes dates de début et de fin d'activité triées par ordre croissant. Clairement, nous avons $Q \leq 2n$.

Par souci de clarté, nous définissons la fonction intermédiaire $\tilde{b}_i(t)$, $\forall i \in \mathcal{A}$, de la façon suivante :

$$\tilde{b}_i(t) = \begin{cases} b_{i0} & \text{si } t \in [t_0, t_1] \\ \quad\vdots & \\ b_{i(Q-1)} & \text{si } t \in [t_{Q-1}, t_Q] \end{cases}$$

avec $b_{iq} = \frac{\int_{t_q}^{t_{q+1}} b_i(t)dt}{t_{q+1} - t_q}$.

La solution $S'$ est alors construite de la manière suivante :

- $st'_i = st_i$
- $et'_i = \min(\tau | \int_{st_i}^{\tau} f_i(\tilde{b}_i(t))dt = W_i)$
- $b'_i(t) = \begin{cases} \tilde{b}_i(t) & \text{si } t \in [st_i, et'_i] \\ 0 & \text{sinon} \end{cases}$

Il est facile de voir que $S'$ satisfait les contraintes de fenêtres de temps (A.1), puisque, par le Lemme 1.1, $et'_i \leq et_i$. De plus, $S'$ vérifie la contrainte d'énergie (A.4) puisqu'elle est définie de cette façon. Enfin, $S'$ vérifie aussi la contrainte de capacité de la ressource (A.5). En effet, comme $S$ est une solution réalisable, nous avons $\forall q \in \{1, \ldots, Q\}$ et $\forall t \in [t_q, t_{q+1}]$ : $\sum_{i\in\mathcal{A}} b_i(t) \leq R \Rightarrow \sum_{i\in\mathcal{A}} \int_{t_q}^{t_{q+1}} b_i(t)dt \leq R(t_{q+1} - t_q)$.

Donc,

$$\begin{aligned}\sum_{i\in\mathcal{A}} b'_i(t) &\leq \sum_{i\in\mathcal{A}} \tilde{b}_i(t)\\ &= \sum_{i\in\mathcal{A}} b_{iq}\\ &= \sum_{i\in\mathcal{A}} \frac{\int_{t_q}^{t_{q+1}} b_i(t)dt}{t_{q+1}-t_q}\\ &\leq R\end{aligned}$$

Nous pouvons montrer que $S'$ vérifie les contraintes de consommation minimale et maximale de la ressource d'une façon similaire. □

Une remarque intéressante peut être faite à partir de la preuve du théorème précédent. En effet, la nouvelle solution $S'$ possède la propriété suivante : l'ensemble des points $t \in \mathcal{H}$ coïncidant avec une variation de la consommation de ressource d'une activité $i$, i.e. $\{t \in \mathcal{H} \mid \forall \epsilon > 0,\ b_i(t) \neq b_i(t+\epsilon)\}$, est contenue dans l'ensemble formé de toutes les dates de début et de fin des activités. C'est ce qu'affirme le corollaire suivant :

**Corollaire 1.3.1** ([NAL15a])**.** $\{t \in \mathcal{H} \mid \forall \epsilon > 0,\ b_i(t) \neq b_i(t+\epsilon)\} \subseteq \{st_i, et_i \mid i \in \mathcal{A}\}$.

De plus, nous pouvons en déduire que le CECSP à dates de début et de fin fixées peut être résolu en temps polynomial.

**Proposition 1.1** ([NAL15b])**.** *Soit* $\Pi$ *une instance du CECSP avec des dates de début,* $st_i$*, et des dates de fin,* $et_i$*, fixées. On peut vérifier que* $\Pi$ *est réalisable en temps polynomial en la taille de l'instance.*

En effet, dans ce cas-là, il suffit de décider pour chaque intervalle composé de deux dates de début/fin consécutives, i.e. de la forme $[st_i, st_j]$, $[st_i, et_j]$, $[et_i, et_j]$ ou $[et_i, st_j]$, la quantité de ressource consommée par chaque activité à l'intérieur de cet intervalle. Ce problème peut facilement être modélisé par un programme linéaire.

Soit $(t_q)_{q=1..Q}$ la suite définie dans la preuve du théorème 1.2 et $b_{iq}$ (respectivement $w_{iq}$), $\forall (i,q) \in \mathcal{A} \times \{1,\ldots,Q-1\}$, la quantité de ressource consommée par (resp. la quantité d'énergie apportée à) l'activité $i$ dans l'intervalle $[t_q, t_{q+1}]$. Rappelons que $Q \leq 2n$. Le programme linéaire s'écrit alors de la manière suivante :

$$\sum_{ß\in A} b_{iq} \leq R(t_{q+1}-t_q) \qquad \forall q \in \{1..Q-1\} \tag{1.13}$$

$$b_{iq} \leq r_i^{max}(t_{q+1}-t_q) \qquad \forall i \in \mathcal{A},\ \forall q \in \{1..Q-1\} |\ t_q \in [st_i, et_i[ \tag{1.14}$$

$$b_{iq} \geq r_i^{min}(t_{q+1}-t_q) \qquad \forall i \in \mathcal{A},\ \forall q \in \{1..Q-1\} |\ t_q \in [st_i, et_i[ \tag{1.15}$$

$$b_{iq} = 0 \qquad \forall i \in \mathcal{A},\ \forall q \in \{1..Q-1\} |\ t_q \notin [st_i, et_i[ \tag{1.16}$$

$$\sum_{q=1}^{Q-1} w_{iq} = W_i \qquad \forall i \in \mathcal{A} \tag{1.17}$$

$$w_{iq} \leq a_{ip} b_{iq} + c_{ip} \qquad \forall i \in \mathcal{A},\ \forall p \in \mathcal{P}_i,\ \forall q \in \{1..Q-1\} \tag{1.18}$$

$$w_{iq} \leq M b_{iq} \qquad \forall i \in \mathcal{A},\ \forall q \in \{1..Q-1\} \tag{1.19}$$

pour $M$ une constante suffisamment grande et $\mathcal{P}_i = \{1, \ldots, P_i\}$ le nombre d'intervalles de définition de la fonction $f_i$. La contrainte (1.13) modélise la contrainte de capacité de la ressource. Les contraintes (1.14) et (1.15) assurent que les contraintes de consommation minimale et maximale de la ressource sont respectées tandis que la contrainte (1.16) fixe la consommation de la ressource à 0 si l'activité n'est pas en cours. La contrainte (1.17) stipule que chaque activité doit recevoir la quantité d'énergie requise. Enfin, les contraintes (1.18) et (1.19) assurent la conversion ressource/énergie. De plus la contrainte (1.19) fixe $w_{iq}$ à 0 si $b_{iq} = 0$, i.e. modélise $f_i(0) = 0$.

On peut remarquer que si $\forall i \in \mathcal{A},\ r_i^{min} = 0$, alors le CECSP devient polynomial. En effet, il suffit de prendre $(t_q)_{q=1..Q}$ la suite des différentes dates de début (resp. fin) au plus tôt (resp. tard). Alors, le programme linéaire précédent nous donne une solution réalisable.

**Théorème 1.4** ([NALR16])**.** *Le CECSP préemptif ($\forall i \in \mathcal{A},\ r_i^{min} = 0$) peut être résolu en temps polynomial.*

De ce fait, dans la suite, nous considérerons que $\exists i \in \mathcal{A}$ tel que $r_i^{min} \neq 0$.

# Conclusion

Dans cette partie, nous avons introduit les principales caractéristiques des problèmes d'ordonnancement avant de nous intéresser en particulier aux problèmes cumulatifs. Nous avons ensuite présenté deux des principaux problèmes étudiés en ordonnancement sous contraintes de ressource : le RCPSP et le CuSP. Les limitations de ces problèmes en termes de modélisation d'activités à profils variables ont ensuite été démontrées et une nouvelle modélisation de la consommation de ressource nous a permis de présenter un nouveau problème : le CECSP.

Dans un premier temps, nous avons comparé ce problème avec les problèmes existant dans la littérature. Cette comparaison nous a permis de montrer que les techniques de résolution existantes ne pouvaient pas s'appliquer directement dans le cadre du CECSP. De plus, le profil variable de consommation de la ressource rend la résolution de ce problème a priori délicate. En effet, dans le cas général, la quantité de ressource allouée à une activité peut varier à tout moment ce qui accroît la difficulté de résolution puisque l'inconnue est une fonction de $\mathbb{R}$ dans $\mathbb{R}$.

Nous avons pu définir un ensemble de propriétés permettant de faciliter le résolution du CECSP. Dans le cas général, ce problème est NP-complet mais il existe des cas particuliers pour lequel ce problème est polynomial. C'est le cas, par exemple, de la version préemptive du CECSP. De plus, nous avons montré que, dans le cas où l'on considère des fonctions de rendement concaves, la fonction d'allocation de ressource est une fonction constante par morceaux et que ses points de rupture correspondent aux dates de début et de fin des activités. Cette propriété nous permet de définir des méthodes de résolution pour le CECSP et aussi d'adapter plusieurs méthodes existantes définies pour d'autres problèmes. Ces méthodes peuvent être classées en deux catégories :

- les techniques adaptées du CuSP et issues de la programmation par contraintes. Ces techniques seront détaillées dans la partie II.
- les techniques adaptées du RCPSP et issues de la programmation linéaire. Ces techniques seront détaillées dans la partie III.

# Deuxième partie

# Programmation par contraintes


**2 Programmation par contraintes et ordonnancement cumulatif** **43**

2.1 La programmation par contraintes . . . . . 43

2.1.1 Problème de satisfaction de contraintes . . . . . 43

2.1.2 Exploration de l'espace de recherche . . . . . 44

2.1.3 Détection d'incohérence et Filtrage . . . . . 46

2.2 L'ordonnancement cumulatif . . . . . 48

2.2.1 L'ordonnancement en programmation par contraintes . . . . . 48

2.2.2 La contrainte cumulative . . . . . 49

2.2.3 Les filtrages de la contrainte cumulative . . . . . 50

**3 Propagation de contraintes pour le CECSP** **67**

3.1 Algorithmes de filtrage basés sur le Time-Table . . . . . 67

3.1.1 Le Time-Table . . . . . 68

3.1.2 Le Time-Table disjonctif . . . . . 69

3.1.3 Le Time-Table basé sur les flots . . . . . 73

3.2 Algorithme de filtrage du raisonnement énergétique . . . . . 75

3.2.1 Algorithme de vérification . . . . . 76

3.2.2 Les ajustements de bornes . . . . . 81

3.2.3 Caractérisation des intervalles d'intérêt . . . . . 85

*Dans cette partie consacrée à l'adaptation des techniques de résolution issues de la programmation par contraintes (PPC), nous commençons par présenter les concepts généraux de ce paradigme. Cette introduction non exhaustive à la programmation par contraintes est suivie d'une description de la modélisation des problèmes d'ordonnancement en PPC.*
*La fin du chapitre 2 présente la contrainte cumulative et quelques-uns des principaux algorithmes de filtrage décrits pour cette contrainte. Pour ces algorithmes de filtrage, nous distinguons trois grands types de raisonnements : les raisonnements simples, ceux utilisant le concept d'énergie et ceux combinant plusieurs raisonnements simples que nous appellerons raisonnements étendus.*
*Enfin, nous montrons comment nous avons adapté plusieurs de ces raisonnements dans le cadre du CECSP dans le chapitre 3. Dans un premier temps, nous présentons l'adaptation de deux raisonnements simples : le Time-Table et le raisonnement disjonctif. Ces deux raisonnements s'adaptent naturellement au cas du CECSP en considérant les activités rectangulaires de durée ou de consommation maximale. Sur le même principe, nous adaptons un des raisonnements étendus récemment proposé pour la contrainte cumulative : le Time-Table disjonctif [GHS15]. Un nouveau raisonnement basé sur le Time-Table et couplé avec un modèle de flots est aussi présenté [NAL15a].*
*La dernière section du chapitre 3 est consacrée à l'adaptation d'un des raisonnements les plus forts définis pour la contrainte cumulative : le raisonnement énergétique. Pour ce raisonnement, nous montrons comment adapter les calculs de consommation minimale dans le cadre du CECSP et qu'il existe un nombre polynomial d'intervalles suffisants pour appliquer le raisonnement de manière complète. De plus, trois méthodes permettant de calculer ces intervalles pour les algorithmes de vérification et d'ajustement sont présentées.*
*Les résultats sur le raisonnement énergétique ont été publiés dans [NAL15a, NAL15b, NALR16].*
*Afin d'assurer la cohérence du manuscrit, certains travaux préliminaires ne sont pas présentés dans ce manuscrit. C'est le cas, par exemple, du modèle de PPC pour le CECSP à temps discret présenté dans [NAL16]. Pour plus de précisions, nous invitons le lecteur à se référer à l'annexe A.*

## Chapitre 2

# Programmation par contraintes et ordonnancement cumulatif

### 2.1 La programmation par contraintes

Cette section s'intéresse à la présentation des concepts de base de la programmation par contraintes (PPC). La programmation par contraintes vise à résoudre des problèmes de satisfaction de contraintes (CSP) mais aussi des problèmes d'optimisation (ce dernier cas ne sera pas traité dans ce manuscrit). Pour cela, un problème est modélisé à l'aide d'un réseau de contraintes et la recherche d'une solution tend à trouver une affectation des variables satisfaisant toutes les contraintes de ce réseau. Une présentation formelle des problèmes de satisfaction de contraintes ainsi qu'un aperçu de quelques méthodes permettant leur résolution sont présentés dans les sous-sections suivantes.

#### 2.1.1 Problème de satisfaction de contraintes

Une instance d'un *problème de satisfaction de contraintes*, ou CSP [1] est la donnée d'un triplet $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ où :

- $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ est l'ensemble des variables du problème ;
- $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$ est l'ensemble des domaines de ces variables, i.e. $x_i \in D_i,\ i = 1, \ldots, n$ ;
- $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ est l'ensemble des contraintes du problème où chaque $c_j$ définit un sous-ensemble du produit cartésien des domaines des variables sur lesquelles elle porte : $c_j(x_{j1}, x_{j2}, \ldots, x_{jk}) \subseteq D_{j1} \times D_{j2} \times \cdots \times D_{jk}$

La notion de domaine désigne l'ensemble des valeurs que peut prendre une variable. La nature de ces domaines peut potentiellement être très différente. Par exemple :

- un ou plusieurs intervalle d'entiers ;
- un ou plusieurs intervalle de réels ;
- un ensemble d'entiers non contigus : il est possible d'utiliser un ensemble d'entiers quelconque, e.g. $D = \{4, 9, 26\}$ ;
- un ensemble de valeurs symboliques : on peut vouloir représenter des couleurs, ou encore des jours de la semaine...

1. en anglais, Constraint Satisfaction Problem.

Une fois les variables définies, nous pouvons ajouter des contraintes les liant entre elles. Formellement, une contrainte peut être définie comme une relation portant sur un ensemble de variables. Ici aussi, plusieurs types de contraintes existent. Nous détaillons trois d'entre elles.

Les premières contraintes présentées sont les contraintes en *extension*. Pour ces contraintes, étant donné un sous-ensemble de variables, on définit explicitement la liste des tuples autorisés. Suivant les cas, ces contraintes peuvent aussi être définies comme une liste de tuples interdits. Les deuxièmes contraintes détaillées sont les contraintes en *intention*. Dans ce cas, chaque contrainte est décrite sous la forme d'une expression arithmétique définissant une relation entre les variables. Enfin, les dernières contraintes décrites sont les contraintes globales. Ces contraintes sont des relations prédéfinies, ayant une signification précise. Des exemples de telles contraintes sont décrits dans l'exemple 2.1.1. Notons que si les variables sont à valeurs dans $\mathbb{R}$ alors une définition en extension de la contraintes peut comprendre un nombre infini de tuples.

**Exemple 2.1.1.** *Soient trois variables* $x_0$, $x_1$ *et* $x_2$ *de domaines respectifs* $D_0 = [0, 2]$, $D_1 = [0, 2]$ *et* $D_2 = [1, 2]$. *Nous considérons les contraintes* $c_0$, $c_1$ *et* $c_2$ *suivantes :*

- $c_0$ *porte sur les variables* $x_0$ *et* $x_1$ *et est décrite en intention :* $x_0 \leq x_1$ ;
- $c_1$ *porte sur* $x_1$ *et* $x_2$ *et est décrite en extension :* $\{(0, 1), (0, 2), (1, 2), (2, 2)\}$ ;
- $c_2$ *est une contrainte globale portant sur les variable* $x_0, x_1$ *et* $x_2$ *:* $allDifferent(x_0, x_1, x_2)$. *Cette contrainte stipule que les valeurs affectées à chaque variable sont différentes les unes des autres.*

*Les domaines des variables étant continus, un nombre infini de tuples aurait été nécessaire pour écrire* $c_0$ *en extension.*

Pour trouver une solution à un CSP, il faut instancier toutes les variables du problème de telle sorte que toutes les contraintes soient satisfaites. Une variable est dite *instanciée* quand on lui assigne une valeur de son domaine. Pour trouver une telle instanciation, un certain nombre de techniques ont été mises en place. Ces techniques reposent principalement sur deux éléments centraux : le filtrage des domaines et l'exploration de l'espace de recherche. Ces deux concepts sont donc décrits dans les sous-sections suivantes.

### 2.1.2 Exploration de l'espace de recherche

Étant donné un CSP, différentes techniques d'exploration de l'espace de recherche peuvent être employées pour trouver des solutions. Cette exploration se fait en général par *séparation et évaluation*, i.e. on sépare le problème difficile à résoudre en deux sous-problèmes, plus petits, jusqu'à obtenir un problème que l'on sera capable de résoudre en temps raisonnable.

Une représentation usuelle du processus de recherche d'une solution est l'*arbre de recherche*. La racine de cet arbre représente le problème que l'on cherche à résoudre et les sommets sont des problèmes réduits, obtenus en décomposant le domaine d'une des variables du problème père. Les feuilles de cet arbre correspondent donc à des instanciations de toutes les variables du problème.

Une première approche consiste alors à générer tous les tuples de valeurs possibles et de tester s'ils sont solution du problème, i.e. si cette instanciation satisfait bien toutes les contraintes du problème. Cela revient à considérer toutes les affectations possibles des variables et ce nombre, qui peut ne pas être fini dans le cas des variables continues, est égal au produit cartésien des cardinaux des domaines

des variables impliquées dans le CSP pour des variables dans $\mathbb{Z}$.

Classiquement, une méthode de séparation, ou de branchement, consiste à fixer une variable à une valeur pour le premier sous-problème et de retirer cette valeur du domaine de la variable dans le second sous-problème. On va alors, à l'aide d'un parcours en profondeur, créer un premier tuple, i.e. une solution candidate, qui pourra ainsi être évalué. Si le tuple satisfait toutes les contraintes du problème, alors c'est une solution et l'algorithme peut s'arrêter. Dans le cas contraire, on évaluera le second sous-problème créé lors de la dernière séparation. Un tel parcours est décrit dans l'exemple 2.1.2.

Dans le cas de CSP comprenant des variables continues, l'énumération de tous les domaines des variables n'est pas possible. Une première approche serait de considérer qu'une variable est instanciée quand son domaine est réduit à un intervalle de taille suffisamment petite. Mais même avec cette restriction supplémentaire, il est très coûteux d'énumérer chacun de ces intervalles. L'idée est alors la suivante : au lieu, à chaque étape de l'algorithme de séparation, d'instancier une variable à une valeur de son domaine, nous séparons le domaine de cette variable en deux (ou plusieurs) sous-domaines.

**Exemple 2.1.2.** *Soient* 3 *variables,* $x_0$, $x_1$ *et* $x_2$, *de domaines respectifs :* $\{1,2\}$, $\{1,2\}$, $\{1,2,3\}$. *Nous considérons les contraintes suivantes :* $c_0 : x_0 < x_1$ *et* $c_1 : x_1 < x_2$.

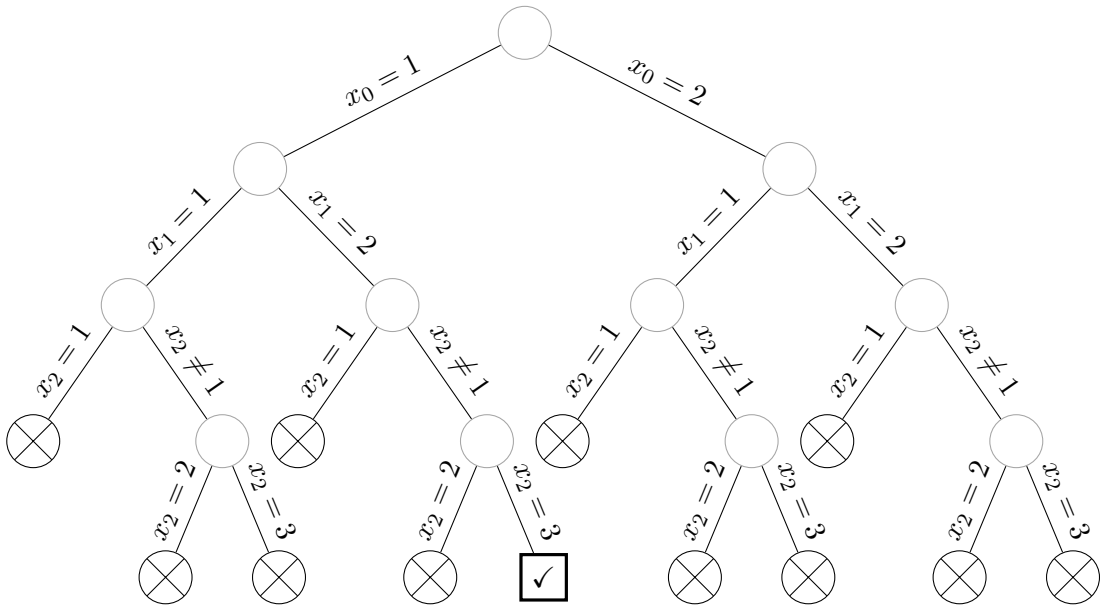



FIGURE 2.1 – Exemple de parcours d'un arbre de recherche.

*La figure 2.1 montre un parcours complet de l'espace de recherche. Les noeuds internes de l'arbre (en gris) sont les noeuds pour lesquels toutes les variables ne sont pas instanciées. Les feuilles de l'arbre correspondent bien à des instanciations complètes des variables. Parmi celles-ci, celles marquées d'une croix sont celles qui ne satisfont pas la contrainte, tandis que celle représentée par un noeud carré correspond à une solution réalisable du problème.*

Ces combinaisons peuvent être générées de différentes manières et donc testées dans des ordres différents. Cet ordre peut avoir une influence déterminante dans le processus de résolution d'un problème. Pour définir cet ordre, on utilise des *heuristiques* de choix de variable et de valeur. Une *heuristique de choix de variable* détermine la variable que l'on va instancier prioritairement. Une *heuristique de*

*choix de valeur* détermine la ou les valeurs à affecter en priorité à cette variable.

Nous donnons ci-après des exemples de telles heuristiques.

**Heuristiques de choix de variable**

- instanciation des variables dans l'ordre lexicographique, c'est l'heuristique utilisée dans la figure 2.1 ;
- instanciation des variables suivant un ordre aléatoire ;
- instanciation de la variable de plus petit domaine [HE80]...

**Heuristiques de choix de valeur**

- sélectionner la valeur minimale (resp. maximale) du domaine de la variable courante, c'est l'heuristique utilisée dans la figure 2.1 ;
- sélectionner une valeur aléatoire dans le domaine de la variable courante ;
- dans le cas de variables continues, on peut choisir de séparer l'intervalle en deux ou plus de morceaux et le séparer au milieu, au tiers...

Dans la suite du manuscrit, nous présenterons d'autres heuristiques que nous appliquerons dans notre processus de recherche.

Même si plusieurs heuristiques efficaces existent, on ne peut être sûr de ne pas devoir explorer tout l'espace de recherche pour trouver une solution. Comme il n'est pas concevable, en pratique, de devoir explorer un tel espace, des techniques permettant de réduire l'arbre de recherche ont donc été mises en place. Les techniques les plus célèbres permettant une telle réduction sont les algorithmes de filtrage. Ces algorithmes sont décrits dans la sous-section suivante.

### 2.1.3 Détection d'incohérence et Filtrage

Afin de réduire l'espace de recherche en programmation par contraintes, il est indispensable de mettre en place des techniques permettant de vérifier la validité d'une solution afin de ne pas considérer un tuple comme étant cohérent s'il ne l'est pas. Ces *algorithmes de détection d'incohérences*, aussi appelés checkers, permettent de refuser une instanciation des variables si cette dernière n'est pas cohérente, i.e. si elle viole un des contraintes. Un algorithme de vérification est donc associé à chaque contrainte.

De plus, ces algorithmes peuvent aussi être étendus de manière à détecter une incohérence en cours de recherche si aucune solution n'est contenue dans le sous-arbre courant.

**Exemple 2.1.3.** *Soit l'instance définie dans l'exemple 2.1.2.*

*La figure 2.2 montre un parcours de l'espace de recherche si l'algorithme utilisé est muni d'un algorithme de détection d'incohérence. Les noeuds noirs correspondent à des noeuds pour lesquels aucune solution ne peut exister dans les sous-arbres ayant un de ces noeuds pour racine. En effet, l'instanciation partielle des variables faites à ce niveau est déjà incohérente avec les contraintes* $c_0$ *(*$x_0 < x_1$*) et* $c_1$ *(*$x_1 < x_2$*).*

Une deuxième technique permettant la réduction de l'espace de recherche est l'utilisation d'algorithme de *filtrage*. Un algorithme de filtrage est aussi défini pour chaque contrainte et cet algorithme

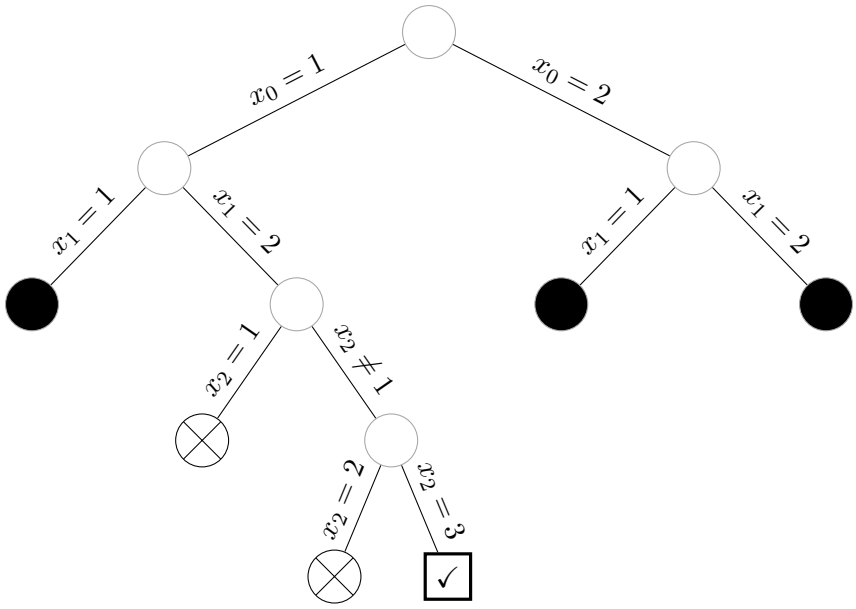



FIGURE 2.2 – Exemple de parcours d'un arbre de recherche avec détection d'incohérences.

consiste à détecter et à retirer des domaines des variables, des valeurs qui ne sont pas cohérentes avec la contrainte.

La mise en place de techniques de filtrage performantes, i.e. qui retirent un maximum de valeurs du domaine des variables, jouent un rôle important dans la résolution de problème en programmation par contraintes. Un exemple simple de filtrage est présenté dans l'exemple 2.1.4.

**Exemple 2.1.4.** *Soit l'instance définie dans l'exemple 2.1.2.*

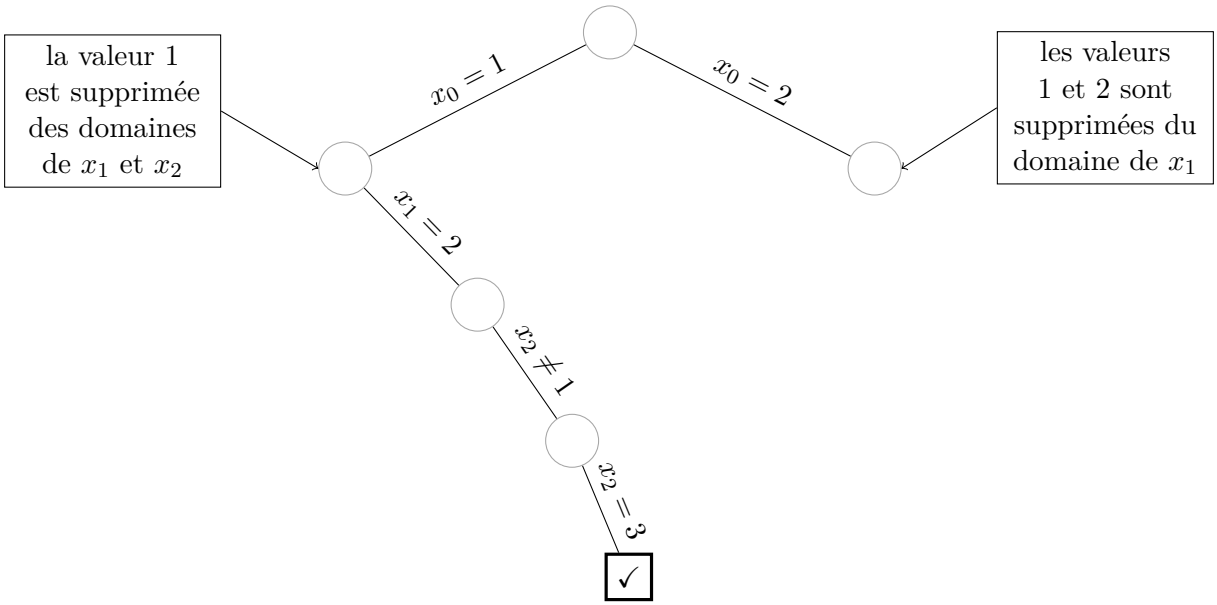



FIGURE 2.3 – Exemple de parcours d'un arbre de recherche avec fitrage des domaines.

*La figure 2.3 montre un parcours de l'espace de recherche si l'algorithme utilisé est muni d'un algorithme de filtrage des domaines. Les contraintes sont considérées une par une. A la racine, on ne peut rien déduire, mais une fois que $x_0$ est instanciée à 1, alors nous pouvons retirer cette valeur du*

*domaine des autres variables car, pour satisfaire les contraintes* $c_0$ *et* $c_1$*, les variables* $x_1$ *et* $x_2$ *doivent être strictement supérieures à* 1*. Puis, on instancie* $x_1$ *à* 2 *et on supprime cette valeur du domaine de* $x_2$*. La seule valeur possible pour* $x_2$ *est alors* 3 *et on obtient une solution.*

*L'autre cas correspond à l'instanciation de* $x_0$ *à* 2*. Dans ce cas-là, les valeurs* 1 *et* 2 *sont rétirées du domaines de* $x_1$ *qui devient vide. Donc il n'existe pas de solution avec* $x_0 = 2$*.*

Dans l'exemple 2.1.4, nous pouvons remarquer que si on avait considéré simultanément les 2 contraintes, on aurait pu déduire la solution au noeud racine.

Quand les contraintes sont binaires, i.e. elles ne concernent que deux variables à la fois, on peut représenter schématiquement le CSP par un graphe. Dans ce graphe, chaque noeud correspond à une variable et chaque arc à une contrainte. Si on vérifie la cohérence des contraintes une par une, on parle alors de cohérence d'arcs. Cette notion peut être généralisée à la considération simultanée de plusieurs contraintes mais aussi à des contraintes contenant plus de deux variables.

Enfin, pour certains problèmes, il n'est pas possible de retirer toutes les valeurs incohérentes du domaines des variables en temps raisonnables. En effet, il peut arriver que les problèmes sous-jacents des contraintes que l'on est amené à résoudre soient NP-complets. Dans ce cas-là, assurer de retirer les valeurs incohérentes signifie que les valeurs restantes sont cohérentes, donc qu'il existe une solution, ce qui ne peut être fait en temps raisonnable puisque le problème est NP-complet. Dans ce cas, il devient crucial de mettre en place des algorithmes qui s'exécutent en temps raisonnable mais qui n'assure pas un filtrage complet du domaine des variables. De tels algorithmes seront présentés dans la sous-section 2.2.3 dans le but de résoudre le CuSP. La section suivante est quant à elle dédiée à la modélisation des problèmes d'ordonnancement en PPC.

## 2.2 L'ordonnancement cumulatif

### 2.2.1 L'ordonnancement en programmation par contraintes

Les problèmes d'ordonnancement étudiés en programmation par contraintes sont majoritairement regroupés en trois catégories : les problèmes préemptifs, les problèmes non préemptifs et les problèmes élastiques. Les problèmes préemptifs (respectivement non préemptifs) sont des problèmes pour lesquels l'interruption des activités est autorisée (resp. non autorisée). Les *problèmes d'ordonnancement élastiques* correspondent aux problèmes où la quantité de ressource attribuée à une activité peut varier, à tout instant $t$ de l'horizon de temps, avec la contrainte que la quantité totale de ressource consommée par l'activité durant son exécution soit égale à une certaine valeur appelée *énergie*. Cette dernière notion peut aisément être étendue dans le cas où l'énergie reçue par une activité n'est pas égale à la quantité de ressource consommée mais où des fonctions de rendement modélisent cette conversion. Clairement, le CECSP est un exemple typique d'un tel problème. Dans la suite de ce chapitre, nous considérons la notion d'énergie au sens de la quantité totale de ressource consommée. L'extension de cette notion pour prendre en considération les fonctions de rendement sera détaillée dans le chapitre 3.

La majorité des problèmes d'ordonnancement non préemptifs classiques peuvent être modélisés à l'aide d'un problème de satisfaction de contraintes. En général, trois variables représentant respectivement la date de début d'une activité, notée $st_i$, sa date de fin, notée $et_i$ et sa durée, notée $p_i$, sont

définies. Les domaines de chacune de ces variables sont définis par les données du problème. En effet, pour chaque activité, nous pouvons calculer une date de début au plus tôt, $est_i$, et au plus tard, $lst_i$ ; ainsi, le domaine de la variable $st_i$ est $[est_i, lst_i]$. De même, le domaine de la variable $et_i$ est $[eet_i, let_i]$, avec $eet_i$ la date de fin au plus tôt de $i$ et $let_i$ sa date de fin au plus tard. La durée d'une activité est quant à elle définie comme la différence entre sa date de fin et sa date de début, i.e. $p_i = et_i - st_i$.

Les problèmes préemptifs sont plus difficiles à modéliser. En effet, dans ce cas-là, un ordonnancement valide ne peut être uniquement représenté par une date de début, de fin et une durée pour chaque activité. Pour ces problèmes, au moins deux modélisations différentes existent. La première consiste à associer à chaque activité une variable d'ensemble, i.e. une variable dont la valeur sera un ensemble. Cette variable représente l'ensemble des instants $t$ où l'activité est en cours, défini comme un ensemble d'intervalles ou de temps $t$ discret. Une seconde possibilité est de définir une variable binaire, pour chaque activité et chaque instant $t$, qui prendra la valeur 1 si l'activité est en cours à l'instant $t$. Notons que, dans le cas d'un problème d'ordonnancement continu, une telle solution ne peut être envisageable car cela conduirait à un nombre infini de telles variables.

Les problèmes élastiques sont, quant à eux, modélisés à travers les contraintes de ressources. Il existe deux principaux types de contraintes de ressource en PPC. Le premier permet de modéliser les ressources disjonctives, i.e. qui ne peuvent exécuter qu'une seule activité à un instant donné, et le second sert à la modélisation des ressources cumulatives. Dans ce manuscrit, nous nous intéressons seulement au second cas.

Étant donné une activité et une ressource de capacité $R$, une variable $b_i$ sert généralement à modéliser la consommation de l'activité sur cette ressource. Dans le cas des tâches élastiques, cette variable est remplacée par une variable $W_i$ représentant l'énergie requise par l'activité. Pour représenter un ordonnancement, un ensemble de variables représentant la quantité de ressource utilisée par une activité à l'instant $t$ est introduit. Ces variables sont ensuite liées entre elles par un ensemble de contraintes modélisant le fait que chaque activité doit recevoir une énergie $W_i$.

Ces différents concepts peuvent être étendus pour modéliser d'autre types de contraintes telles que des contraintes de ressources alternatives, des temps de préparation entre les activités, des activités optionnelles ou des contraintes de réservoirs.

Dans la suite de ce manuscrit, nous nous intéressons aux problèmes d'ordonnancement non préemptifs avec ressource cumulative et aux problèmes d'ordonnancement élastiques. Le premier cas correspond au CuSP, décrit dans la sous-section 1.1.2. En PPC, le problème est modélisé à l'aide d'une contrainte globale appelée contrainte cumulative. Cette contrainte ainsi que différents algorithmes de filtrage mis en place pour cette dernière sont présentés dans les deux sous-sections suivantes.

Les problèmes d'ordonnancement élastiques seront représentés par le CECSP (voir section 1.2), pour lequel l'adaptation de certains algorithmes de filtrage pour la contrainte cumulative est présentée dans le chapitre 3.

### 2.2.2 La contrainte cumulative

En PPC, le CuSP est modélisé par la contrainte cumulative. Dans ce contexte, une activité est souvent représentée par 4 variables : $st_i$, $et_i$, $p_i$ et $b_i$ correspondant respectivement à la date de

début de l'activité, sa date de fin, sa durée et sa consommation de ressource. En règle générale, les domaines des variables $p_i$ et $b_i$ sont restreints à un seul élément, i.e. la durée de l'activité est fixe et sa consommation de ressource est constante durant toute son exécution. Les domaines des variables $st_i$ et $et_i$ sont quant à eux considérés finis.

La contrainte cumulative vise donc à déterminer seulement les dates de début et de fin des activités – liés par la condition d'intégrité $et_i = st_i + p_i$. Cette contrainte prend donc en paramètres l'ensemble d'activités $\mathcal{A}$ à ordonnancer et la capacité $R$ de la ressource utilisée pour exécuter les activités. Avec ces notations la contrainte s'écrit $cumulative(\mathcal{A}, R)$ et cette contrainte est satisfaite si et seulement si :

$$\forall i \in \mathcal{A}, \quad st_i + p_i = et_i \tag{2.1}$$

$$\forall t \in \mathcal{H}, \quad \sum_{\substack{i \in A \\ t \in [st_i, et_i[}} b_i \leq R \tag{1.3}$$

La seconde contrainte ainsi qu'un exemple de solution sont détaillés dans la sous-section 1.1.2.

L'existence d'une solution au problème cumulatif étant un problème NP-complet, un algorithme effectuant un filtrage complet des domaines des variables, i.e. retirant toutes les valeurs incohérentes des domaines des variables, ne peut s'exécuter en temps polynomial. De plus, assurer seulement la cohérence des bornes des domaines étant aussi NP-complet, les algorithmes utilisés permettent seulement d'ajuster certaines bornes des domaines en fonction des données et contraintes du problème sans s'assurer de la cohérence de toutes les bornes. Ces algorithmes de filtrage se présentent le plus souvent sous la forme de règles, appelées *règles d'ajustement*, et sont appliqués sur des relaxations de la contraintes cumulative et des règles pouvant être appliquées en temps polynomial ont été mises en place pour des relaxations.

Dans ce manuscrit, nous présentons quelques une de ces techniques. Pour une description plus détaillée des différentes techniques de relaxation existantes pour la contrainte cumulative, nous renvoyons le lecteur à [BLPN01, DHP99]. Baptiste *et al.* [BLPN01] présentent un panorama de ces différentes relaxations ainsi que les principales techniques issues de la programmation par contraintes mises en place pour résoudre la contrainte cumulative. Dans un contexte plus général, Dorndoff *et al.* [DHP99] exhibent des règles de cohérence basées sur la capacité des intervalles, i.e. la quantité de ressource disponible dans un intervalle.

La sous-section suivante présente plusieurs de ces règles d'ajustement définies pour la contrainte cumulative.

### 2.2.3 Les filtrages de la contrainte cumulative

Le problème cumulatif étant un problème symétrique, les règles d'ajustement sont souvent définies pour une seule des deux variables $st_i$ et $et_i$. En effet, une fois ces règles définies pour une de ces deux variables, les règles symétriques peuvent être déduites pour l'autre variable. Les raisonnements et algorithmes décrits dans cette sous-section ne présentent que le filtrage des dates de début au plus tôt.

Dans un premier temps, nous présentons des règles simples permettant l'ajustement des bornes des

variables. Ensuite, une partie sera consacrée aux règles d'ajustement basées sur le concept d'énergie. Enfin, ces règles seront combinées afin de créer des algorithmes permettant un filtrage plus fort des domaines des variables.

Plusieurs des règles présentées dans cette sous-section seront ensuite adaptées au cas du CECSP dans le chapitre 3.

### Règles de filtrages simples

Parmi les raisonnements les plus simples appliqués dans le cadre de la contrainte cumulative, on retrouve le raisonnement basé sur l'échéancier (on utilise le terme anglais de Time-Table) [Lah82] et le raisonnement disjonctif [BLPN01].

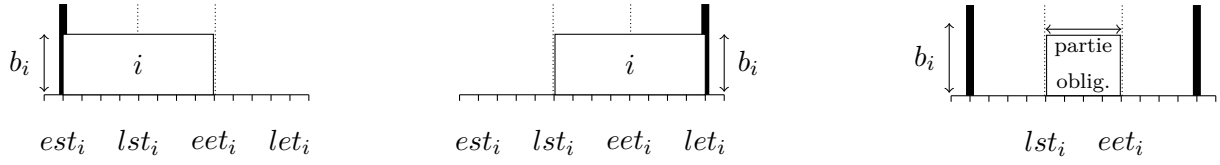
**Time-Table** Le premier algorithme de filtrage de la contrainte cumulative présenté est basé sur la notion de *partie obligatoire* d'une activité. Cette partie obligatoire correspond à l'intervalle de temps maximal pendant lequel on est sûr que l'activité devra être exécutée. Cet intervalle est vide dans le cas où $lst_i \geq eet_i$, et est égal à $[lst_i, eet_i[$ dans le cas contraire. Notons que dans le cas de la contrainte cumulative, une borne supérieure sur la date de début d'une activité peut aisément être calculée. En effet, il suffit de prendre $lst_i = let_i - p_i$.

**Définition 2.1.** *La partie obligatoire d'une activité $i$ est définie par l'intervalle $[lst_i, eet_i[$.*

**Exemple 2.2.1.** *Considérons l'activité suivante :*

| $est_i$ | $let_i$ | $p_i$ | $b_i$ |
|---|---|---|---|
| 1 | 13 | 8 | 2 |

*Nous pouvons calculer sa date de début au plus tard, $lst_i = let_i - p_i = 13 - 8 = 5$, ainsi que sa date de fin au plus tôt, $eet_i = est_i + p_i = 1 + 8 = 9$. Comme $eet_i > lst_i$, l'activité possède une partie obligatoire qui est l'intervalle $[5, 9[$ (voir figure 2.4).*




(a) Ordonnancement au plus tôt (b) Ordonnancement au plus tard (c) Partie obligatoire

FIGURE 2.4 – Partie obligatoire d'une activité $i$ pour le CuSP.

Les parties obligatoires des activités peuvent ensuite être agrégées de manière à obtenir une fonction en escalier, appelée *profil obligatoire de la ressource.*

**Définition 2.2.** *Le* **profil obligatoire** $TT_{\mathcal{A}}$ *d'une ressource est défini par la fonction suivante :* $TT_{\mathcal{A}}(t) = \sum_{\substack{i \in \mathcal{A} \\ lst_i \leq t < eet_i}} b_i, \ \forall t \in \mathcal{H}$. *Le problème n'admet pas de solution si* $\exists t \in \mathcal{H} \ : \ TT_{\mathcal{A}}(t) > R$.

**Exemple 2.2.2.** *Dans l'exemple décrit par la figure 2.5, nous remarquons que, au temps $t = 1$, le profil obligatoire de la ressource vaut 2. Comme la ressource est de capacité 5, aucune incohérence n'est détectée. À l'inverse, au temps $t = 5$, le profil obligatoire de la ressource a une valeur de 6, ce*

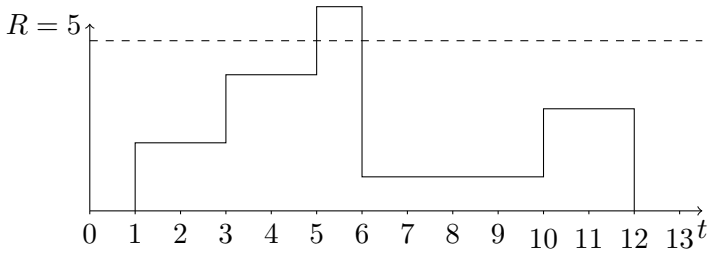



FIGURE 2.5 – Exemple de profil obligatoire d'une ressource cumulative.

*qui est supérieur à la capacité de la ressource. Une incohérence est donc détectée et il n'existe pas de solution réalisable pour cette instance.*

Pour réaliser un filtrage des bornes, nous devons fournir des règles permettant d'identifier si débuter l'activité $i$ à $est_i$ (ou $lst_i$) ne viole pas les contraintes du problème. Cette règle peut être formalisée de la façon suivante :

**Règle 2.1.** *Soit une activité $i \in \mathcal{A}$. S'il existe $t \in [est_i, lst_i[$ tel que $t < eet_i$ et que $b_i + TT_{\mathcal{A}\setminus\{i\}}(t) > R$ alors la date de début au plus tôt de $i$ peut être ajustée et on a : $est_i > t$.*

**Exemple 2.2.3.** *Soit l'activité définie ci-dessous.*

| $est_i$ | $let_i$ | $p_i$ | $b_i$ |
|---|---|---|---|
| 1 | 14 | 8 | 2 |

*La figure 2.6 présente le profil obligatoire de la ressource ainsi que l'ajustement déduit de la règle 2.1. Dans cet exemple, on voit bien que l'activité ne peut pas commencer à tout temps $t \in [est_i, 6[$. En effet,*

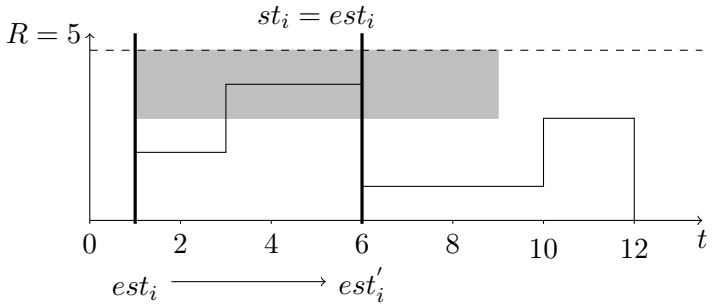



FIGURE 2.6 – Règle d'ajustement du Time-Table pour le CuSP.

*le profil obligatoire de la ressource montre que la quantité de ressource disponible dans l'intervalle $[3, 6[$ ne permet pas d'exécuter l'activité pendant cet intervalle. La date de début au plus tôt peut donc être mise à jour ($est'_i = 6$).*

Beldiceanu *et al.* [BC02] ont introduit un algorithme de balayage permettant d'appliquer les ajustements décrits ci-dessus en $O(n^2)$. D'autres algorithmes ont aussi été mis en place pour ce raisonnement [OQ13, LBC12, GHS15a]. Ces algorithmes possèdent des complexités théoriques moins élevées

ou équivalentes à celle de l'algorithme de balayage de Beldiceanu *et al.* en proposant des filtrages moins importants.

**Raisonnement disjonctif** Une deuxième règle permettant d'ajuster les bornes des variables repose sur un raisonnement appelé le *raisonnement disjonctif*. Ce raisonnement est brièvement décrit dans [BLPN01] et présenté plus en détail dans [GHS15].

Ce raisonnement repose sur le concept d'ensembles disjonctifs. Ces ensembles correspondent aux ensembles d'activités qui ne peuvent s'exécuter en parallèle. Son application la plus basique s'intéresse seulement aux ensembles de taille 2, c'est-à-dire aux paires d'activités disjonctives.

Si nous considérons deux activités $(i, j) \in \mathcal{A}^2$ telles que $b_i + b_j > R$ alors une des deux affirmations suivantes doit être vérifiée :

— l'activité $i$ doit commencer après la fin de l'activité $j$ ;

— l'activité $i$ doit finir avant le début de l'activité $j$.

En effet, comme les activités $i$ et $j$ ne peuvent être exécutées en parallèle, l'une doit forcément finir avant que l'autre ne puisse s'exécuter.

Cette propriété nous permet d'améliorer les bornes des variables de début d'activité selon la règle suivante :

**Règle 2.2.** *Soient $i,\ j \in \mathcal{A}$, $i \neq j$ telles que $b_i + b_j > R$ et $lst_i < eet_j$. Alors la date de début au plus tôt de l'activité peut être ajustée et on a : $est_j \geq eet_i$.*

**Exemple 2.2.4.** *Soient $i$ et $j$ les deux activités suivantes :*

| $act$ | $est_i$ | $let_i$ | $p_i$ | $b_i$ |
|---|---|---|---|---|
| $i$ | 2 | 11 | 4 | 2 |
| $j$ | 1 | 20 | 7 | 2 |

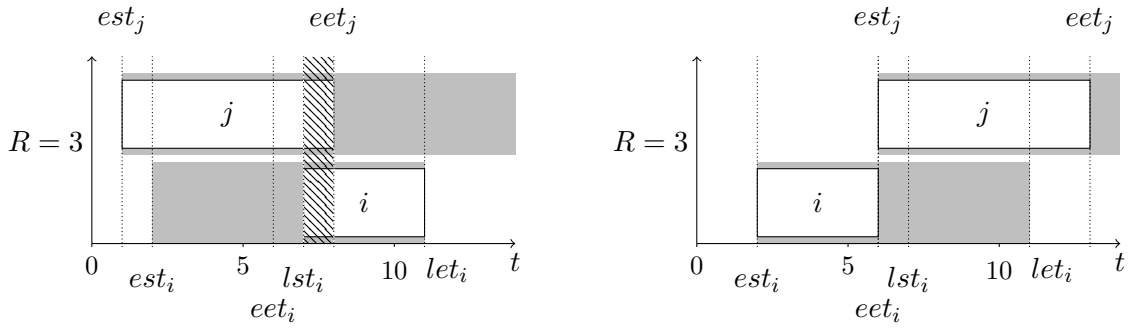



FIGURE 2.7 – Raisonnement disjonctif pour le CuSP.

*Dans l'exemple ci-dessus, nous avons $b_i + b_j = 4 > 3$. Donc $i$ et $j$ ne peuvent s'exécuter en parallèle. Comme $eet_j = 8 > 7 = lst_i$, $i$ doit être exécutée avant $j$. En effet, $i$ doit forcément démarrer avant $lst_i = 7$ et ,si $j$ est ordonnancée au plus tôt, i.e. $st_j = est_j = 1$, $j$ ne peut finir avant $eet_j = 8$. Donc, dans ce cas, $i$ et $j$ se chevauchent. L'activité $j$ ne peut commencer à $est_j$ et doit commencer après $eet_i = 6$. Donc $est_j$ est ajustée à cette valeur.*

L'exemple 2.2.4 montre que, dans certains cas, le raisonnement disjonctif est plus fort que le Time-Table. En effet, dans cet exemple, aucune des activités ne possède de partie obligatoire. Aucun ajustement n'est donc détecté par la règle 2.1.

À l'inverse, dans certains cas, le raisonnement Time-Table va procéder à plus d'ajustements que le raisonnement disjonctif. En effet, le raisonnement disjonctif présenté ici ne raisonne que sur des paires d'activités tandis que le Time-Table est capable de détecter des ajustements déduits de n-uplets d'activités. Il faut cependant que les activités possèdent une partie obligatoire. Cependant, le Time-Table est souvent plus utilisé en pratique car il peut amener les mêmes déductions que les paires de disjonction tout en ayant une plus faible complexité.

Dans la sous-section 2.2.3, nous montrerons comme il est possible de coupler ces deux raisonnements pour en obtenir un nouveau, le Time-Table disjonctif [GHS15].

Le paragraphe suivant est dédié aux règles de filtrage utilisant le concept d'*énergie*.

#### Règles de filtrage énergétiques

Parmi les règles de filtrage utilisant le concept d'énergie on trouve le Edge-Finding [Vil09a, Nui94], le raisonnement énergétique [JE89] ou encore les activités élastiques [BLPN01]. Nous présenterons en détail le second cas, tandis que les autres cas seront brièvement discutés. La raison principale étant que nous adapterons le raisonnement énergétique dans le cas du CECSP.

L'idée principale des raisonnements énergétique est de comparer la quantité de ressource disponible dans un intervalle avec l'énergie que doit consommer un ensemble de tâches à l'intérieur de ce même intervalle. Les raisonnements diffèrent principalement sur la manière de calculer cette énergie et sur les intervalles considérés.

Le raisonnement Edge-Finding s'applique sur un sous-ensemble d'activités $\Omega \subseteq \mathcal{A}$, le but étant de décider si une activité $j$ doit obligatoirement commencer avant $\Omega$ dans toute solution réalisable. Dans ce cas, l'énergie d'une activité $i$ représente sa charge de travail $W_i$ et est simplement exprimée comme $W_i = p_i \times b_i$. L'énergie nécessaire pour un ensemble d'activités $\Omega$ est alors $\sum_{i\in\Omega} W_i$. Les intervalles considérés dans ce raisonnement correspondent aux fenêtres de temps des activités $[est_i, let_i[$. Cette notion est étendue à $\Omega$ de la façon suivante : $[est_\Omega, let_\Omega[ = [\min_{j\in\Omega} est_j, \max_{j\in\Omega} let_j[$. L'idée du raisonnement est alors de vérifier que la ressource disponible permet de faire débuter l'activité $i$ dans la fenêtre de temps correspondant à $\Omega$. Pour cela, l'énergie requise par $\Omega \cup \{i\}$ est comparée à la quantité de ressource disponible dans l'intervalle $[est_\Omega, let_{\Omega\cup\{i\}}[$. Si la quantité de ressource n'est pas suffisante, alors l'activité $i$ doit commencer avant $est_\Omega$ et un ajustement peut être effectué.

Dans le cas des activités partiellement et totalement élastiques, les contraintes de consommation de ressource sont relâchées et une activité n'est plus contrainte de consommer une quantité constante de la ressource au cours du temps. Selon que l'activité est totalement ou partiellement élastique, ces contraintes sont plus ou moins relâchées et la consommation de ressource des activités peut subir des variations plus ou moins grandes au cours du temps. L'avantage de ces relaxations est que l'existence d'une solution peut être décidée en temps polynomial. La façon dont les ajustements sont calculés étant très similaire à celle dont ils sont calculés dans le cadre du raisonnement énergétique, ils ne sont pas détaillés dans ce manuscrit. Une description de ces ajustements peut être trouvé dans [BLPN99].

Le raisonnement énergétique, introduit par [JE89], est un des raisonnements les plus forts existant pour la contrainte cumulative. Le principe est de considérer un intervalle $[t_1, t_2[$ et de calculer l'énergie requise par une activité à l'intérieur de cet intervalle. Dans ce cas, l'énergie requise par une activité dans un intervalle $[t_1, t_2[$ équivaut à la quantité minimale de ressource qu'elle doit utiliser à l'intérieur de cet intervalle. Cette quantité, notée $\underline{b}(i, t_1, t_2)$, est ensuite utilisée pour calculer la fonction de marge $SL(t_1, t_2)$ correspondant à la différence entre la quantité de ressource requise par toutes les activités à l'intérieur de $[t_1, t_2[$ et la quantité de ressource disponible dans ce même intervalle, $R(t_2 - t_1)$. Si cette quantité est négative pour un intervalle $[t_1, t_2[$, alors une incohérence est détectée. En effet, si c'est le cas, alors cela signifie que la quantité de ressource disponible dans $[t_1, t_2[$ n'est pas suffisante pour ordonnancer les consommations minimales de toutes les activités.

**Définition 2.3.** *La fonction de marge $SL(t_1, t_2)$ est définie de la façon suivante :*

$$SL(t_1, t_2) = R * (t_2 - t_1) - \sum_{i \in \mathcal{A}} \underline{b}(i, t_1, t_2)$$

Avec cette définition, nous pouvons maintenant énoncer le théorème central du raisonnement énergétique :

**Théorème 2.1.** *Soit $\mathcal{I}$ une instance de la contrainte cumulative. Alors, nous avons la propriété suivante :*

$$\exists t_1, t_2 \in \mathbb{N}, t_1 < t_2, SL(t_1, t_2) < 0 \Rightarrow \mathcal{I} \text{ ne possède pas de solution} \tag{2.2}$$

Pour calculer cette fonction de marge, il faut pouvoir calculer la quantité de ressource requise par une activité dans l'intervalle $[t_1, t_2[$, $\underline{b}(i, t_1, t_2)$. Étant donné une activité $i$, cette quantité correspond à l'aire minimale, parmi tous les ordonnancements possibles de $i$, occupée par cette dernière à l'intérieur de l'intervalle $[t_1, t_2[$. Pour atteindre ce minimum, seules trois configurations sont possibles et sont décrites ci-dessous :

— l'activité est *calée à gauche* : l'activité démarre à $est_i$ ;
— l'activité est *calée à droite* : l'activité finit à $let_i$ ;
— l'activité est *centrée* : elle occupe tout l'intervalle $[t_1, t_2[$.

Parmi ces trois cas, celui conduisant à la consommation minimale de ressource est celui dont l'intersection avec l'intervalle $[t_1, t_2[$ est minimale.

Pour donner l'expression mathématique de ces quantités, nous introduisons trois notations. $\underline{b}_{LS}(i, t_1, t_2)$ (respectivement $\underline{b}_{RS}(i, t_1, t_2)$ et $\underline{b}_{CS}(i, t_1, t_2)$) correspond à la quantité de ressource requise par l'activité $i$ dans l'intervalle $[t_1, t_2[$ quand l'activité est calée à gauche (respectivement calée à droite et centrée). Formellement, ces trois quantités peuvent être exprimées de la manière suivante :

$$\underline{b}_{LS}(i, t_1, t_2) = b_i * \max(0, p_i - \max(0, t_1 - est_i)) \tag{2.3}$$
$$\underline{b}_{RS}(i, t_1, t_2) = b_i * \max(0, p_i - \max(0, let_i - t_2)) \tag{2.4}$$
$$\underline{b}_{CS}(i, t_1, t_2) = b_i * (t_2 - t_1) \tag{2.5}$$

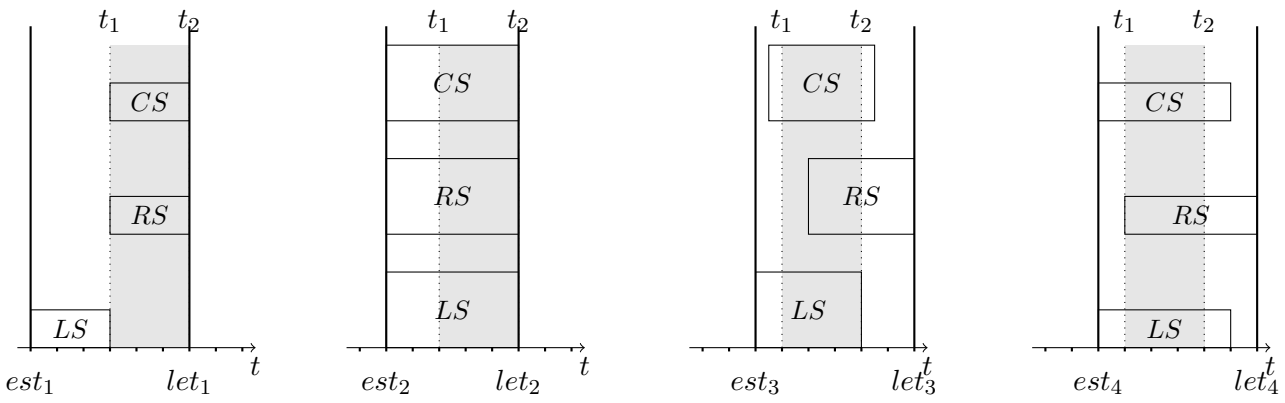
Alors, l'expression de la quantité minimale de ressource est le minimum de ces trois quantités, i.e.

$$\underline{b}(i, t_1, t_2) = \min\left(\underline{b}_{LS}(i, t_1, t_2), \underline{b}_{RS}(i, t_1, t_2), \underline{b}_{CS}(i, t_1, t_2)\right) \tag{2.6}$$

**Exemple 2.2.5.** *Considérons une ressource de capacité $R = 4$ et les 4 activités suivantes :*

| $act$ | $est_i$ | $let_i$ | $p_i$ | $b_i$ |
|---|---|---|---|---|
| 1 | 0 | 6 | 3 | 1 |
| 2 | 1 | 6 | 5 | 2 |
| 3 | 2 | 8 | 4 | 2 |
| 4 | 2 | 8 | 5 | 1 |

*Nous montrons comment appliquer le raisonnement énergétique sur l'intervalle $[3, 6[$. Nous commençons par présenter le calcul de la consommation minimale de chaque activité. Dans la figure 2.10, nous représentons graphiquement les activités calées à gauche (LS), à droite (RS) et centrées (CS).*




(a) activité 1 (b) activité 2 (c) activité 3 (d) activité 4

FIGURE 2.8 – Calcul de l'énergie minimale dans un intervalle $[t_1, t_2[$ pour le CuSP.

*Le tableau suivant présente le calcul de l'énergie minimale de chaque activité. Ces valeurs sont calculées à partir des équations (2.3), (2.4) et (2.5).*

| $act$ | $\underline{b}_{LS}(i, 3, 6)$ | $\underline{b}_{RS}(i, 3, 6)$ | $\underline{b}_{CS}(i, 3, 6)$ | $\underline{b}(i, 3, 6)$ |
|---|---|---|---|---|
| 1 | $1 * (3 - (3 - 0)) = 0$ | $1 * (3 - (6 - 3)) = 3$ | $1 * (6 - 3) = 3$ | 0 |
| 2 | $2 * (5 - (3 - 1)) = 6$ | $2 * (5 - (6 - 6)) = 10$ | $2 * (6 - 3) = 6$ | 6 |
| 3 | $2 * (4 - (3 - 2)) = 6$ | $2 * (4 - (8 - 6)) = 4$ | $2 * (6 - 3) = 6$ | 4 |
| 4 | $1 * (5 - (3 - 2)) = 3$ | $1 * (4 - (8 - 6)) = 3$ | $1 * (6 - 3) = 3$ | 3 |

*La valeur de la fonction de marge $SL(3, 6)$ est donc :*

$$\Big(4 * (6 - 3)\Big) - \Big(0 + 6 + 4 + 3\Big) = 12 - 13 = -1 < 0$$

*L'instance n'est donc pas réalisable et une incohérence est détectée.*

La règle de détection d'incohérence décrite par le Théorème 2.1 impose le calcul de la fonction de marge pour tout intervalle $[t_1, t_2[ \in \mathbb{N}^2$, ce qui, en pratique, n'est pas envisageable. Cependant, Baptiste *et al.* ont proposé une caractérisation des intervalles sur lesquels il est suffisant d'appliquer

le raisonnement énergétique. Par suffisant, nous entendons que, si une incohérence n'est pas détectée sur cet ensemble d'intervalles, alors aucun autre intervalle ne permettra de conclure à l'insatisfiabilité de l'instance. La taille de cet ensemble d'intervalles d'*intérêt* est de l'ordre de $O(n^2)$. Le calcul de la fonction de marge pouvant s'effectuer de manière incrémentale, la complexité du checker énergétique est donc de $O(n^2)$.

Récemment, Derrien *et al.* [DP14] ont proposé une caractérisation plus fine de l'ensemble des intervalles d'intérêt pour le raisonnement énergétique. L'ordre de grandeur de la taille de cet ensemble d'intervalles n'est pas modifié et est toujours de l'ordre de $n^2$ mais le nombre d'intervalles considérés est divisé par 7. En effet, la caractérisation de Baptiste *et al.* considère, pour chaque paire d'activités, 15 intervalles tandis que la caractérisation de Derrien *et al.* n'en considère que 2.

Ces caractérisations se basent toutes les deux sur l'idée suivante : pour décider de l'existence d'un point $(t_1, t_2)$ pour lequel la fonction de marge évaluée en ce point $SL(t_1, t_2)$ est négative, il suffit d'évaluer la fonction en ses minima locaux. La principale différence entre ces deux caractérisations est la caractérisation plus ou moins fine de ces minima.

Nous allons brièvement détailler la caractérisation de Derrien *et al.* que nous adapterons dans le cadre du CECSP dans le chapitre 3.

Pour étudier les minima locaux de la fonction de marge, les auteurs de [DP14] étudient les variations de cette dernière. L'expression de la fonction de marge dépendant de $R*(t_2-t_1)$ et de $\sum_{i\in\mathcal{A}} \underline{b}(i,t_1,t_2)$, ses variations dépendent des variations de la fonction $(t_1,t_2) \to \sum_{i\in\mathcal{A}} \underline{b}(i,t_1,t_2)$. En particulier, ils remarquent que tous les changements de variation de cette fonction ne peuvent impliquer un minimum local. Pour une fonction quelconque, une variation implique un tel minimum seulement si la dérivée à gauche de la fonction est négative, sa dérivée à droite est positive et au moins l'une d'entre elles est non nulle.

Cette propriété, induite par le *test de la dérivée seconde* et appliquée au cas du problème cumulatif, leur permet de décrire des conditions sur les consommations individuelles des activités, $\underline{b}(i,t_1,t_2)$, sous lesquelles le point $(t_1,t_2)$ peut être un minimum local de la fonction de marge. Ces conditions sont exprimées dans le lemme 2.1.

**Lemme 2.1.** *Si le point $(t_1,t_2)$ est un minimum local de la fonction de marge $SL(t_1,t_2)$, alors il existe deux activités $i$ et $j$ telles que les conditions ci-dessous sont satisfaites.*

$$\frac{\delta^- \underline{b}(i,t_1,t_2)}{\delta t_1} > \frac{\delta^+ \underline{b}(i,t_1,t_2)}{\delta t_1} \tag{2.7}$$

$$\frac{\delta^- \underline{b}(j,t_1,t_2)}{\delta t_2} > \frac{\delta^+ \underline{b}(j,t_1,t_2)}{\delta t_2} \tag{2.8}$$

*avec $\frac{\delta^+ \underline{b}(j,t_1,t_2)}{\delta t_2}$ (resp. $\frac{\delta^- \underline{b}(j,t_1,t_2)}{\delta t_2}$) la dérivée à droite (resp. gauche) de $t_2 \to \underline{b}(j,t_1,t_2)$.*

*Démonstration.* Soit $(t_1,t_2) \in \mathbb{N}^2$ tel qu'il n'existe aucune activité $i$ vérifiant la condition (2.7). Alors, $\sum_{i\in\mathcal{A}} \underline{b}(i,t_1,t_2)$ a sa dérivée à gauche inférieure ou égale à sa dérivée à droite. Par le test de la dérivée seconde, un minimum local d'une fonction ne peut exister que si la dérivée à gauche est plus grande que la dérivée à droite. Donc le point $(t_1,t_2)$ ne peut pas être un minimum local.

De la même façon, on peut prouver que si la condition (2.8) n'est pas vérifiée, alors $(t_1, t_2)$ ne peut pas être un minimum local. □

Le lemme 2.1 peut ensuite être utilisé pour établir les conditions nécessaires permettant de déterminer un ensemble des intervalles d'intérêt. Pour cela, une étude des fonctions $t_1 \to \underline{b}(i, t_1, t_2)$ et $t_2 \to \underline{b}(i, t_1, t_2)$ est nécessaire. Nous montrons comment obtenir une caractérisation des différentes dates de début possibles pour un intervalle, la caractérisation des dates de fin pouvant s'obtenir de manière similaire.

**Lemme 2.2.** *Pour chaque activité $i$ et pour tout début d'intervalle $t_1$ il existe au plus un intervalle $[t_1, t_2[$ tel que $\frac{\delta^- \underline{b}(i,t_1,t_2)}{\delta t_2} > \frac{\delta^+ \underline{b}(i,t_1,t_2)}{\delta t_2}$ :*

1. *si $t_1 \leq est_i$ alors seulement l'intervalle $[t_1, let_i[$ doit être considéré.*
2. *si $t_1 > lst_i \wedge t_1 \geq eet_i$ alors aucun intervalle ne doit être considéré.*
3. *si $t_1 > lst_i \wedge t_1 < eet_i \wedge t_1 < lst_i$ alors seulement l'intervalle $[t_1, est_i + let_i - t_1[$ doit être considéré.*
4. *si $t_1 > lst_i \wedge t_1 < eet_i \wedge t_1 \geq lst_i$ alors seulement l'intervalle $[t_1, eet_i[$ doit être considéré.*

*Démonstration.* Pour prouver ce lemme, nous étudions les variations de la fonction $t_2 \to \underline{b}(i, t_1, t_2)$, à $t_1$ fixé. Les variations de cette fonction dépendent de la position relative de $t_1$ par rapport à $est_i$, $lst_i$ et $eet_i$. Le cas où $t_1 \leq est_i$ est étudié ci-dessous et illustré pour une activité $i$ possédant les caractéristiques suivantes : $est_i = 2$, $let_i = 8$, $p_i = 4$ et $b_i = 1$.

- si $t_2 \leq lst_i$ alors $\underline{b}(i, t_1, t_2) = 0$
- si $lst_i < t_2 < let_i$ alors $\underline{b}(i, t_1, t_2) = b_i(t_2 - lst_i)$
- si $let_i \leq t_2$ alors $\underline{b}(i, t_1, t_2) = p_i$

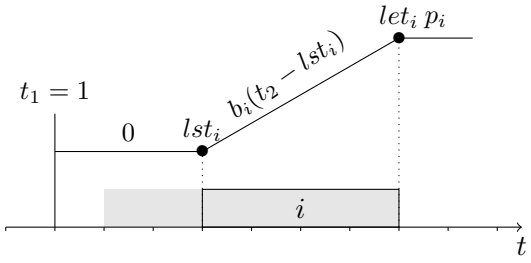



FIGURE 2.9 – Évolution de la fonction de consommation minimale en fonction de $t_2$ pour le CuSP.

Le seul point $t_2$ pour lequel $\frac{\delta^- \underline{b}(i,t_1,t_2)}{\delta t_2} > \frac{\delta^+ \underline{b}(i,t_1,t_2)}{\delta t_2}$ est $[1, 8[ = [t_1, let_i[$.

Les trois autres cas peuvent être prouvés de façon très similaire. □

Un raisonnement symétrique permet de caractériser l'ensemble des points $t_1$ vérifiant la condition (2.7). Une fois que ces dates potentielles de début et de fin d'intervalle d'intérêt sont caractérisées, elles sont rassemblées pour créer l'ensemble des intervalles d'intérêt pour le raisonnement énergétique. La caractérisation complète de ces intervalles est décrite dans le lemme suivant.

**Lemme 2.3.** *Si le point $(t_1, t_2)$ est un minimum local de la fonction de marge $SL(t_1, t_2)$, alors il existe deux activités $i$ et $j$ telles que $[t_1, t_2[ = \mathcal{O}_C(i, j)$ avec :*

$$\mathcal{O}_C(i,j) = \begin{cases} [est_i, let_j[ & si \quad est_i \leq est_j \ \wedge \ let_j \geq let_i \\ [est_i, est_j + let_j - est_i[ & si \quad est_i > est_j \ \wedge \ est_i > eet_j \ \wedge \\ & \quad est_j + let_j - est_i \geq let_i \\ [est_i, eet_j[ & si \quad est_i > est_j \ \wedge \ est_i < eet_j \ \wedge \ est_i \geq lst_j \ \wedge \\ & \quad eet_j \geq let_i \\ [lst_i, let_j[ & si \quad lst_i \leq est_j \ \wedge \ let_j < let_i \ \wedge \ let_j > lst_i \ \wedge \\ & \quad let_j \leq eet_j \\ [lst_i, est_j + let_j - lst_i[ & si \quad lst_i > est_j \ \wedge \ let_i < eet_j \ \wedge \ lst_i < lst_j \ \wedge \\ & \quad lst_i < est_j + let_j - lst_i \ \wedge \ est_j + let_j - lst_i < let_i \\ [lst_i, eet_j[ & si \quad lst_i > est_j \ \wedge \ lst_i < eet_j \ \wedge \ lst_i \geq lst_j \ \wedge \\ & \quad eet_j < let_i \ \wedge \ eet_j > lst_i \ \wedge \ eet_j \leq eet_i \\ [est_i + let_i - let_j, let_j[ & si \quad let_j < let_i \ \wedge \ let_j > lst_i \ \wedge \ let_j > eet_i \ \wedge \\ & \quad est_i + let_i - let_j \leq lst_j \\ [est_i + let_i - eet_j, eet_j[ & si \quad eet_j < let_i \ \wedge \ eet_j > lst_i \ \wedge \ eet_j > eet_i \ \wedge \\ & \quad lst_j \leq est_i + let_i - eet_j < eet_j \ \wedge \\ & \quad lst_j < est_i + let_i - eet_j \end{cases}$$

Une démonstration de ce lemme ainsi que la preuve que ces intervalles sont suffisants pour détecter toute incohérence due au raisonnement énergétique peuvent être trouvées dans [Der15].

Grâce à cette caractérisation, l'algorithme de détection d'incohérence de Baptiste *et al.* peut être amélioré. Pour cela, nous considérons l'ensemble $\mathcal{O}_1 = \cup_{i \in \mathcal{A}} \{\langle i, est_i \rangle, \langle i, lst_i \rangle\}$ des dates de début possibles d'un intervalle et l'ensemble $\mathcal{O}_2 = \cup_{i \in \mathcal{A}} \{\langle i, eet_i \rangle, \langle i, let_i \rangle\}$ correspondant aux dates de fin. De plus, nous définissons l'ensemble $O_t = \cup_{i \in \mathcal{A}} \{\langle i, est_i + let_i - t \rangle\}$. Ceci nous permet de décrire l'algorithme (cf. 1) de détection d'incohérence en $O(n^2)$ de Derrien [Der15].

**Algorithme 1 :** Algorithme de détection d'incohérence énergétique pour le CuSP.

**pour tous les** $\langle i, t_1 \rangle \in \mathcal{O}_1$ **faire**
- $pente = \sum_j \underline{b}(j, t_1, t_1 + 1)$
- $SL_{ER} = 0$, $t_2^{old} = t_1$
- **pour tous les** $\langle j, t_2 \rangle \in \mathcal{O}_2 \cup \mathcal{O}_{t_1}$ *par ordre croissant* **faire**
  - $SL_{ER} = SL_{ER} + pente * (t_2 - t_2^{old})$
  - **si** $SL_{ER} > R * (t_2 - t_1)$ **alors**
    - Le problème est incohérent.
  - $pente = pente + \underline{b}(j, t_1, t_2 - 1) - 2\underline{b}(j, t_1, t_2) + \underline{b}(j, t_1, t_2 + 1)$
  - $t_2^{old} = t_2$

Cette caractérisation peut aussi être étendue pour déterminer les intervalles $(t_1, t_2)$ sur lesquels

appliquer les ajustements du raisonnement énergétique. Ces ajustements sont décrits par les règles 2.3 et 2.4. Dans la règle 2.3, nous imposons à une activité $i$ de commencer après $t_1$ et si la quantité de ressource disponible n'est pas suffisante pour ordonnancer les consommations minimales de toutes les activités – excepté celle de l'activité $i$ qui est remplacée par $\underline{b}_{RS}(i, t_1, t_2)$ – alors, nous pouvons déduire que l'activité doit commencer avant $t_1$.

**Règle 2.3.** *S'il existe un intervalle* $[t_1, t_2[$ *avec* $t_1 > est_i$ *et une activité* $i$ *pour lesquels :*

$$\sum_{\substack{j \in \mathcal{A} \\ j \neq i}} \underline{b}(j, t_1, t_2) + \underline{b}_{RS}(i, t_1, t_2) > R(t_2 - t_1)$$

*alors*

$$lst_i \leq t_1 - \frac{1}{b_i} \left( \sum_{\substack{j \in \mathcal{A} \\ j \neq i}} \underline{b}(j, t_1, t_2) + \underline{b}_{RS}(i, t_1, t_2) - R(t_2 - t_1) \right)$$

**Exemple 2.2.6.** *Considérons une ressource de capacité* $R = 3$ *et les* 4 *activités suivantes :*

| $act$ | $est_i$ | $let_i$ | $p_i$ | $b_i$ |
|---|---|---|---|---|
| 1 | 0 | 6 | 5 | 2 |
| 2 | 2 | 6 | 1 | 3 |
| 3 | 2 | 5 | 3 | 1 |
| 4 | 0 | 4 | 2 | 1 |

*Pour pouvoir ajuster la date de début au plus tard de l'activité* 4*, nous allons appliquer la règle 2.3 sur l'intervalle* $[2, 4[$*. Pour cela, nous commençons par calculer la quantité de ressource requise par les activités dans cet intervalle.*

*Pour l'activité* 1*, cette quantité est* $\underline{b}(1, 2, 4) = 4$*. Elle est atteinte en calant l'activité à gauche ou à droite (l'énergie est la même dans les deux cas). Pour l'activité* 2*, en calant l'activité à droite, nous obtenons* $\underline{b}(2, 2, 4) = 0$*. Enfin, pour l'activité* 3*, la quantité de ressource requise vaut* $\underline{b}(3, 2, 4) = 2$*.*

*La quantité de ressource obtenue en calant l'activité* 4 *à droite est de* $\underline{b}_{RS}(4, 2, 4) = 2$*. Nous devons ensuite comparer la somme de toutes ces consommations, i.e.* $4 + 0 + 2 + 2 = 8$*, à la quantité de ressource disponible dans l'intervalle* $[2, 4[$*, i.e.* $R(t_2 - t_1) = 3 * (4 - 2) = 6$*. Comme nous avons* $6 < 8$*, nous pouvons appliquer la règle 2.3 et nous obtenons* $lst_4 \leq 2 - (6 + 2 - 6)/1 = 0$*. Le domaine de la date de début de l'activité* 4 *est donc réduit à* $\{0\}$*.*


(a) l'activité 4 ne peut commencer après $t_1 = 2$

(b) le domaine de l'activité 4 peut être ajusté

FIGURE 2.10 – Ajustement de borne du raisonnement énergétique pour le CuSP.

Le même raisonnement que pour l'algorithme de détection d'incohérence nous permet de caractériser l'ensemble des intervalles permettant de réaliser des ajustements. Les conditions sous lesquelles un intervalle peut conduire à un ajustement sont décrites dans le lemme 2.4. Comment obtenir cette caractérisation à partir de ce lemme sera aussi adapté dans le cadre du CECSP. De ce fait, elle n'est pas décrite ici.

**Lemme 2.4.** *Si le point* $(t_1, t_2)$ *est un minimum local de la fonction* $R(t_2 - t_1) - \sum_{\substack{j \in \mathcal{A} \\ j \neq i}} \underline{b}(j, t_1, t_2) - \underline{b}_{RS}(i, t_1, t_2)$*, alors une des conditions ci-dessous est satisfaite.*

$$\exists (j,k),\ \frac{\delta^- \underline{b}(j,t_1,t_2)}{\delta t_1} > \frac{\delta^+ \underline{b}(j,t_1,t_2)}{\delta t_1} \wedge \frac{\delta^- \underline{b}(k,t_1,t_2)}{\delta t_2} > \frac{\delta^+ \underline{b}(k,t_1,t_2)}{\delta t_2} \tag{2.9}$$

$$\exists j,\ \frac{\delta^- \underline{b}(j,t_1,t_2)}{\delta t_1} > \frac{\delta^+ \underline{b}(j,t_1,t_2)}{\delta t_1} \wedge \frac{\delta^- \underline{b}_{RS}(i,t_1,t_2)}{\delta t_2} > \frac{\delta^+ \underline{b}_{RS}(i,t_1,t_2)}{\delta t_2} \tag{2.10}$$

$$\exists k,\ \frac{\delta^- \underline{b}_{RS}(i,t_1,t_2)}{\delta t_1} > \frac{\delta^+ \underline{b}_{RS}(i,t_1,t_2)}{\delta t_1} \wedge \frac{\delta^- \underline{b}(k,t_1,t_2)}{\delta t_2} > \frac{\delta^+ \underline{b}(k,t_1,t_2)}{\delta t_2} \tag{2.11}$$

$$\frac{\delta^- \underline{b}_{RS}(i,t_1,t_2)}{\delta t_1} > \frac{\delta^+ \underline{b}_{RS}(i,t_1,t_2)}{\delta t_1} \wedge \frac{\delta^- \underline{b}_{RS}(i,t_1,t_2)}{\delta t_2} > \frac{\delta^+ \underline{b}_{RS}(i,t_1,t_2)}{\delta t_2} \tag{2.12}$$

L'ordre de grandeur de l'ensemble des intervalles d'intérêt pour les ajustements est aussi de $n^2$ et l'algorithme permettant de procéder à tous les ajustements possibles a une complexité en $O(n^3)$. Ceci fait du raisonnement énergétique un des algorithmes de filtrage le moins utilisé en pratique, même si ce raisonnement est un des plus forts pour la contrainte cumulative.

De manière similaire à la règle 2.3, une autre règle permettant d'ajuster la date de début au plus tôt d'une activité peut être déduite.

**Règle 2.4.** *S'il existe un intervalle* $[t_1, t_2[$ *avec* $t_1 > est_i$ *et une activité* $i$ *pour lesquels :*

$$\sum_{\substack{j \in \mathcal{A} \\ j \neq i}} \underline{b}(j, t_1, t_2) + \min(\underline{b}_{CS}(i, t_1, t_2), \underline{b}_{LS}(i, t_1, t_2)) > R(t_2 - t_1)$$

*alors*

$$est_i \geq t_2 - \frac{1}{b_i} \left( R(t_2 - t_1) - \sum_{\substack{j \in \mathcal{A} \\ j \neq i}} \underline{b}(j, t_1, t_2) \right)$$

Le lemme 2.4 peut aussi être appliqué dans ce cas, en considérant la fonction suivante au lieu de $\underline{b}_{RS}(i, t_1, t_2)$ : $b_i * \max\Big(0\,,\, \min(\,let_i\,,\,t_2\,) - \max(\,lst_i\,,\,t_1\,)\Big)$.

D'autres algorithmes ont été mis en place pour améliorer le raisonnement énergétique. Ces derniers possèdent souvent une complexité théorique moins élevée que celle de l'algorithme de Derrien *et al.* mais les ajustements appliqués par ces derniers sont souvent incomplets [Tes16, BHS11, Bon16].

Les différentes règles de filtrages présentées dans les paragraphes précédents ont parfois été étendues ou couplées afin de définir de nouvelles règles plus fortes que considérées indépendamment. Certaines d'entre elles sont décrites dans le paragraphe suivant.

### Règles de filtrage étendues

Parmi les règles de filtrage étendues ou couplées on trouve le Edge-Finding étendu [MVH08], le Time-Table-Edge-Finding [Vil11] ou encore le Time-Table disjonctif [GHS15]. Dans ce paragraphe, nous montrerons comment coupler deux raisonnements en présentant, en premier lieu, l'idée de la méthode utilisée pour le Time-Table-Edge-Finding et, dans un second temps, en détaillant celle utilisée pour le Time-Table disjonctif. En effet, une adaptation sera présentée dans le chapitre 3 dans le cadre du CECSP.

L'idée du Time-Table-Edge-Finding est de renforcer le raisonnement existant du Edge-Finding en prenant en considération le profil obligatoire de la ressource. L'algorithme proposé par Vilím est en $O(n^2)$ et est un des plus performants en pratique pour résoudre les problèmes cumulatifs.

Afin d'ajouter l'énergie déduite du raisonnement Time-Table à celle du raisonnement Edge-Finding, les activités sont séparées en deux parties, la partie obligatoire et la partie libre. Ceci permet d'éviter de comptabiliser l'énergie deux fois lors du calcul de cette dernière. L'énergie consommée par un ensemble d'activités dans un intervalle est alors égale à la somme des énergies des parties libres de ces dernières et de la fonction de profil obligatoire de la ressource à l'intérieur de cet intervalle.

Dans le même esprit, Gay [GHS15] propose de combiner le raisonnement disjonctif et le raisonnement Time-Table. Pour ce faire, l'auteur définit la notion d'*intervalle minimum de superposition*. Cette notion va permettre de définir une nouvelle règle de filtrage dans le cas où faire débuter une activité $j$ à sa date de début au plus tôt $est_j$ implique que cette activité chevauchera forcément une activité $i$ vérifiant $b_i + b_j > R$ dans tout ordonnancement réalisable. Ceci est dû au fait que la fenêtre de temps de $j$ ne peut contenir un point que l'activité $i$ devra forcément chevaucher. Quand l'activité $i$ n'a pas de partie obligatoire, l'intervalle que $j$ ne peut intersecter peut être caractérisé.

**Définition 2.4.** *L'intervalle minimal de superposition d'une activité $i$, noté $moi_i$, est l'intervalle de temps le plus petit tel que $i$ s'exécute au moins durant un point de temps de cet intervalle, et ce indépendamment du moment auquel l'activité $i$ est exécutée.*

*Formellement, l'intervalle minimal de superposition est défini par* $[eet_i - 1, lst_i]$.

Quand une activité possède une partie obligatoire, elle ne possède pas d'intervalle minimum de superposition.

**Exemple 2.2.7.** *La figure 2.11 illustre l'intervalle minimum de superposition d'une activité $i$ ne possédant pas de partie obligatoire.*

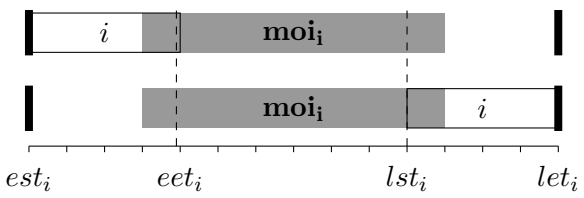



FIGURE 2.11 – Intervalle minimum de superposition d'une activité pour le CuSP.

Cette notion permet de définir une première règle d'ajustement. Cette règle sera améliorée dans un second temps.

**Règle 2.5.** *Soient deux activités $i$ et $j$ telles que $i$ ne possède pas de partie obligatoire et $b_i + b_j > R$. Si ordonnancer l'activité $j$ à sa date de début au plus tôt la fait se superposer complètement à l'intervalle minimal de superposition de $i$ ($moi_i \subseteq [est_j, eet_j]$), alors $est_j \geq eet_i$.*

**Exemple 2.2.8.** *La règle 2.5 est illustrée par la figure 2.12. Dans la partie gauche de la figure, on peut constater que les activités $i$ et $j$ ne peuvent être exécutées en parallèle à cause de la capacité de la ressource. Si l'activité $j$ commence à $est_j$, alors elle intersecterait complètement $moi_i$ et il serait impossible d'ordonnancer $i$. Sur la partie droite de la figure, $est_j$ a été ajusté et l'activité $j$ ne peut commencer avant* $\min\{t \in \mathcal{H} | t \in moi_i\}$ *($\mathcal{H}$ étant l'horizon de temps du problème).*

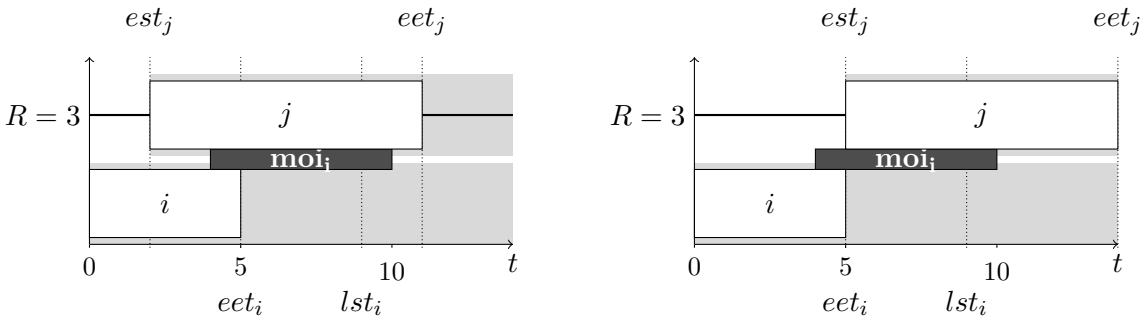



FIGURE 2.12 – Illustration de la règle 2.5 pour le CuSP.

Nous montrons maintenant comment les auteurs de [GHS15] intègrent les informations déduites du Time-Table pour proposer une règle de filtrage plus forte que celle décrite ci-dessus. Cette règle sera décrite seulement dans le cas où les activités $i$ et $j$ ne possèdent pas de partie obligatoire. Le cas inverse est décrit dans [GHS15] et repose sur la séparation des activités en une partie obligatoire et une partie libre.

La règle 2.5 compare seulement les consommations de $i$ et de $j$ avec la capacité de la ressource $R$. Cependant, les consommations obligatoires des autres activités peuvent ne pas laisser $R$ unités de ressource durant l'intersection de $i$ et de $j$. La règle suivante prend donc en considération le profil obligatoire de la ressource.

**Règle 2.6.** *Soient $i$ et $j$ deux activités qui ne possèdent pas de partie obligatoire et telles que $b_i + b_j + \min_{t \in moi_i} TT_{\mathcal{A}}(t) > R$. Si ordonnancer l'activité $j$ à sa date de début au plus tôt la fait se superposer complètement à l'intervalle minimal de superposition de $i$ ($moi_i \subseteq [est_j, eet_j]$), alors $est_j \geq eet_i$.*

**Exemple 2.2.9.** *Considérons les activités suivantes :*

| $act$ | $est_i$ | $let_i$ | $p_i$ | $b_i$ |
|---|---|---|---|---|
| 1 | 2 | 11 | 3 | 2 |
| 2 | 1 | 20 | 9 | 1 |
| 3 | 2 | 11 | 9 | 1 |

*Les activités 1 et 2 ne possèdent pas de partie obligatoire tandis que l'activité 3 est forcément en cours d'exécution durant l'intervalle* $[2, 11[$.

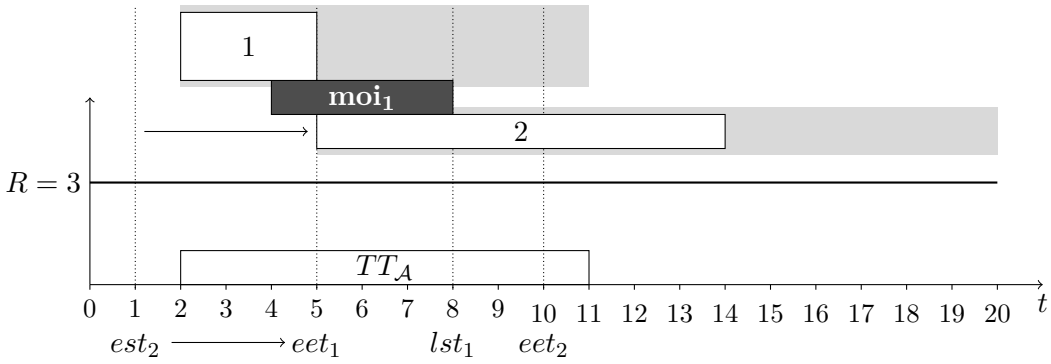



FIGURE 2.13 – Illustration du Time-Table disjonctif dans le cadre du CuSP.

*L'intervalle* $moi_1 = [4, 9[$ *est complètement inclus dans l'intervalle formé par* $est_2$ *et* $eet_2$*, i.e.* $[1, 10[$*. De plus, le minimum du profil de consommation de la ressource dans* $moi_1$ *est de* 1*. Donc, les activités* 1 *et* 2*, consommant respectivement* 2 *et* 1 *unités de ressource, ne peuvent se chevaucher sur l'intervalle* $[4, 9[$*. Donc* $est_2$ *peut être ajusté à* 5*.*

On peut remarquer qu'aucun des autres raisonnements présentés dans ce chapitre n'aurait filtré de valeurs du domaine de $st_2$. En particulier, le raisonnement énergétique ne détecte pas d'ajustement dans ce cas. Le raisonnement Time-Table disjonctif n'est donc dominé par aucun autre raisonnement existant.

## Conclusion

Dans ce chapitre, nous avons d'abord présenté les bases de la programmation par contraintes. En particulier, nous avons vu que les problèmes d'ordonnancement se traduisent naturellement en problème de satisfaction de contraintes.

Ce paradigme est ensuite appliqué au CuSP, un des plus célèbres problèmes d'ordonnancement traité en PPC. De ce fait, nous avons pu présenter de nombreux algorithmes de filtrage, i.e. des algorithmes filtrant les valeurs incohérentes des domaines des variables, pour ce problème. Ces algorithmes, issus de la littérature, sont classés en trois classes : les algorithmes de filtrage simples, ceux basés sur le concept d'énergie et les algorithmes de filtrage étendus. Pour chacune de ces trois classes, au moins un algorithme de propagation est présenté en détail et de nombreux autres exemples sont cités.

Dans le paragraphe concernant les règles de filtrages simples, nous avons présenté le raisonnement Time-Table et le raisonnement disjonctif. Ces raisonnements sont parmi les moins forts pour le CuSP mais leur facilité d'implémentation et leur rapidité en font des algorithmes très utilisés en pratique.

Le paragraphe suivant a présenté les règles de filtrage énergétiques. Parmi elles, sont mentionnés le raisonnement Edge-Finding, les activités élastiques et le raisonnement énergétique qui est présenté en détail. Ce dernier est, à ce jour, un des raisonnements les plus forts pour le CuSP. Cependant, sa complexité élevée fait de lui un des algorithmes les moins utilisés en pratique.

Enfin, le dernier paragraphe explique comment certains auteurs ont étendu ou couplé des règles de filtrage existantes pour définir de nouvelles règles de filtrage plus fortes que prises individuellement.

Parmi celles citées, on trouve le raisonnement Edge-Fiding étendu, le Time-Table-Edge-Finding et le Time-Table disjonctif. A l'inverse du raisonnement énergétique, ces règles sont très utilisées en pratique car leur complexité est relativement faible.

Dans le prochain chapitre, nous présenterons l'adaptation de plusieurs de ces raisonnements dans le cadre du CECSP. Un nouveau raisonnement, basé sur le Time-Table, sera aussi présenté.

## Chapitre 3

# Propagation de contraintes pour le CECSP

Dans ce chapitre, nous décrivons des algorithmes et modèles issus de la programmation par contraintes. Les deux sections suivantes sont dédiées à la présentation d'algorithmes de filtrage pour le problème de décision du CECSP. En effet, comme dans le cas du CuSP, la NP-complétude de ce problème implique qu'un algorithme assurant la cohérence des bornes de chaque variable – garantissant l'existence d'une solution où les valeurs de chaque variable sont comprises dans l'intervalles défini par ces bornes – ne peut s'exécuter en temps polynomial. Par conséquent, plusieurs relaxations sont proposées afin de supprimer les valeurs possibles de début et fin de tâches incohérentes en temps polynomial.

La section 3.1 est dédiée aux extensions pour le CECSP des raisonnements pour le CuSP basés sur le Time-Table (voir chapitre précédent). Dans un premier temps, nous montrons qu'une partie des raisonnements pour le CuSP s'adapte directement au cas du CECSP. En effet, dans le CECSP, une activité peut être vue comme deux sous-activités :

— l'une correspondant à une activité rectangulaire, appelée *partie fixe*, dont les dates de début et de fin doivent être déterminées et consommant une quantité $r_i^{min}$ de la ressource ;
— la seconde, appelée *partie malléable*, correspondant à une activité interruptible de même date de début que la partie minimale et dont chaque partie consomme une quantité de ressource comprise entre 0 et $r_i^{max} - r_i^{min}$.

La présence d'activités rectangulaires permet alors l'adaptation directe de certains raisonnements existants pour le CuSP. Ici, nous présenterons les adaptations des raisonnements Time-Table (3.1.1), disjonctif et Time-Table disjonctif (3.1.2).

Dans un second temps (3.1.3), nous présenterons un nouveau raisonnement étendu, couplant Time-Table et problème de flots.

Enfin, une adaptation complète du raisonnement énergétique est décrite dans la section 3.2.

### 3.1 Algorithmes de filtrage basés sur le Time-Table

La section suivante présente plusieurs algorithmes de filtrage pour le CECSP. Tous ces algorithmes utilisant une adaptation du Time-Table pour le CuSP, nous commençons donc par présenter brièvement comment ce raisonnement est modifié pour être adapté dans le cadre du CECSP. Puis, nous présentons deux autres algorithmes permettant de réduire l'ensemble des valeurs possibles pouvant être prises par chaque variable. Le premier est adapté du Time-Table disjonctif pour le CuSP [GHS15] et le dernier

utilise une combinaison entre problème de flots et profil obligatoire.

### 3.1.1 Le Time-Table

Comme pour le CuSP, le Time-Table pour le CECSP se base sur la notion de partie obligatoire des activités, i.e. l'intervalle pendant lequel une activité est en cours d'exécution dans tous les ordonnancements réalisables. Cependant, comme dans le cas du CECSP nous ne connaissons pas la durée exacte d'une activité, nous utilisons une borne inférieure sur sa durée pour calculer la date de début au plus tard, $lst_i$, et la date de fin au plus tôt, $eet_i$. Pour calculer cette borne, remarquons que la configuration permettant de finir une activité le plus rapidement possible, est celle où l'activité est exécutée à son rendement maximal $r_i^{max}$. De ce fait, une borne inférieure sur la durée de l'activité vaut $W_i/f_i(r_i^{max})$. Nous pouvons donc calculer la date de début au plus tard de l'activité, $lst_i = let_i - W_i/f_i(r_i^{max})$, et sa date de fin au plus tôt, $eet_i = est_i + W_i/f_i(r_i^{max})$. La partie obligatoire d'une activité $i$ est alors définie de la même manière que pour le CuSP, i.e la partie obligatoire de $i$ est l'intervalle $[lst_i, eet_i[$ (cf. figure 3.1).

Cependant, dans le cas où $lst_i \leq eet_i$, i.e. où l'activité possède une partie obligatoire, nous pouvons seulement déduire que l'activité $i$ va consommer au moins une quantité $r_i^{min}$ de la ressource durant toute sa partie obligatoire, et ce, quel que soit le moment où l'activité est ordonnancée. La notion de profil obligatoire de la ressource est donc légèrement différente de celle définie pour le CuSP (cf. définition 2.2, page 51).

**Définition 3.1.** *Le profil obligatoire d'une ressource $TT_{\mathcal{A}}$ dans le cas du CECSP est défini de la façon suivante :*

$$TT_{\mathcal{A}}(t) = \sum_{\substack{i \in \mathcal{A} \\ lst_i \leq t \leq eet_i}} r_i^{min} \quad \forall t \in \mathcal{H}$$

*Le problème est donc insatisfiable dans le cas où $\exists t \in \mathcal{H} \ : \ TT_{\mathcal{A}}(t) > R$*

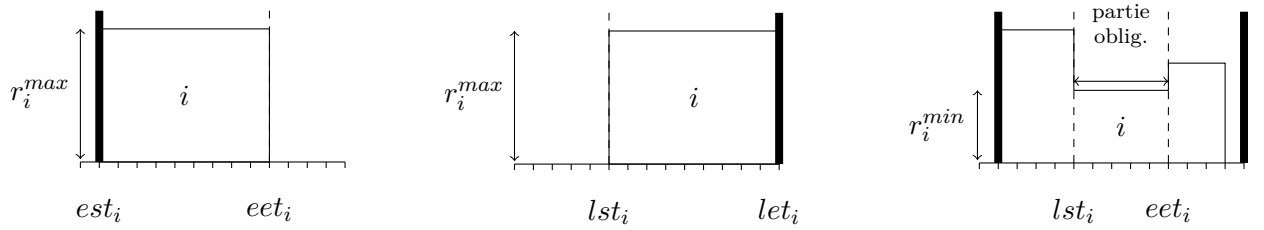
**Exemple 3.1.1.** *Considérons l'activité suivante :*

| $est_i$ | $let_i$ | $W_i$ | $r_i^{min}$ | $r_i^{max}$ | $f_i(b_i(t))$ |
|---|---|---|---|---|---|
| 1 | 14 | 72 | 2 | 5 | $b_i(t) + 3$ |

*Nous pouvons calculer sa date de début au plus tard, $lst_i = let_i - W_i/f_i(r_i^{max}) = 14 - 72/8 = 5$, ainsi que sa date de fin au plus tôt, $eet_i = est_i + W_i/f_i(r_i^{max}) = 1 + 72/8 = 10$. Comme $eet_i > lst_i$, l'activité possède une partie obligatoire qui est l'intervalle $[5, 10[$ (voir figure 3.1a, 3.1b). Cependant, nous pouvons seulement en déduire que l'activité sera en cours dans cet intervalle et, grâce à la borne inférieure sur la quantité de ressource que peut consommer l'activité durant son exécution, nous pouvons déduire que, sur l'intervalle $[lst_i, eet_i[ = [5, 10[$, l'activité est au moins exécutée à $r_i^{min}$ (voir figure 3.1c).*

Comme pour la contrainte cumulative, le profil obligatoire peut aussi être calculé en $O(n)$ à l'aide d'un algorithme de balayage en triant au préalable les activités par date de début au plus tard et date de fin au plus tôt.

Nous détaillons maintenant l'adaptation du Time-Table disjonctif au CECSP.




(a) Ordonnancement au plus tôt

(b) Ordonnancement au plus tard

(c) Ordonnancement réalisable à $r_i^{min}$ dans $[lst_i, eet_i]$

FIGURE 3.1 – Partie obligatoire d'une activité $i$ pour le CECSP.

### 3.1.2 Le Time-Table disjonctif

Le second algorithme de filtrage proposé repose sur un raisonnement appelé Time-Table disjonctif et utilisé, en premier lieu, pour le CuSP (cf. sous-section 2.2.3). Ce dernier repose sur le raisonnement Time-Table décrit précédemment et sur le raisonnement disjonctif.

Le raisonnement disjonctif dans le cadre du CECSP est très similaire à celui défini pour le CuSP. La différence repose sur la construction des ensembles disjonctifs. Dans le cas du CECSP, un couple d'activités $(i, j)$ sera dit disjonctif si $r_i^{min} + r_j^{min} > R$. Dans ce cas, nous savons que :

— l'activité $i$ doit commencer après l'activité $j$, ou

— l'activité $i$ doit finir avant l'activité $j$.

Cette propriété permet notamment d'ajuster la date de début au plus tôt de d'une des deux activités. Considérons par exemple le cas où $est_j \leq eet_i$ et $lst_i < eet_j$ ; ces relations impliquent que $j$ doit commencer après la fin de l'activité $i$ (voir figure 3.3). De ce fait, le début de $j$ ne peut arriver avant la date de fin au plus tôt de $i$, et donc : $est_j \geq eet_i$. La règle de filtrage est ensuite similaire à celle mise en place pour le CuSP.

**Règle 3.1.** *Soient $i, j \in \mathcal{A}, i \neq j$ telles que $r_i^{min} + r_j^{min} > R$ et $lst_i < eet_j$. Alors la date de début au plus tôt de l'activité peut être ajustée et on a : $est_j \geq eet_i$.*

**Exemple 3.1.2.** *Considérons l'instance à deux activités et avec une ressource de capacité* 3 *suivante :*

| $act$ | $est_\ell$ | $let_\ell$ | $W_\ell$ | $r_\ell^{min}$ | $r_\ell^{max}$ | $f_\ell(b_\ell(t))$ |
|---|---|---|---|---|---|---|
| $i$ | 2 | 11 | 28 | 2 | 3 | $2 * b_i(t) + 1$ |
| $j$ | 1 | 20 | 49 | 2 | 4 | $voir fig.$ 3.2 |

*La fonction $f_j(b_j(t))$ est définie par l'expression suivante :*

$$f_j(b_j(t)) = \begin{cases} 2b_j(t) & b_j(t) \in [2,3] \\ b_j(t) + 3 & b_j(t) \in [3,4] \end{cases}$$

*et décrite dans la figure 3.2*

*Dans cet exemple, comme $r_i^{min} + r_j^{min} = 4 > 3$, $i$ et $j$ ne peuvent s'exécuter en parallèle. Si l'activité $i$ finit au temps $let_i = 11$ alors, elle chevauche forcément l'activité $j$ (voir figure 3.3a et 3.3b). Dans tout ordonnancement réalisable, $i$ est donc exécutée avant $j$ et la date de début au plus tôt de $j$ peut donc être ajustée, i.e. $j$ ne peut commencer avant $eet_i = 6$ (voir figure 3.3c).*

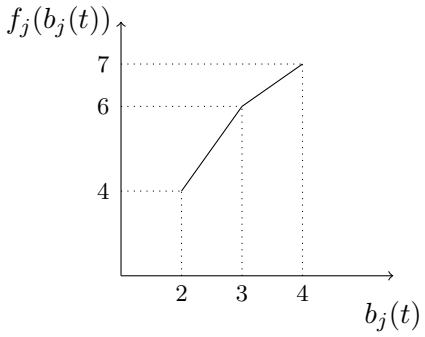
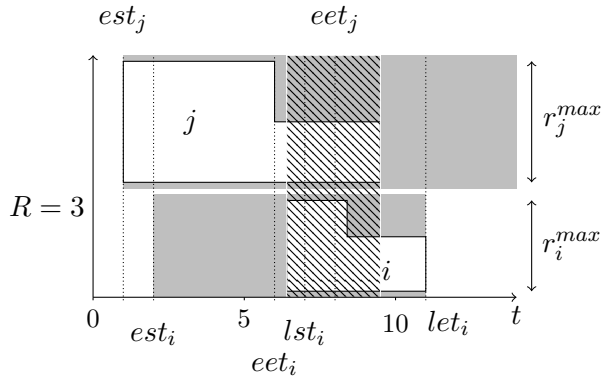
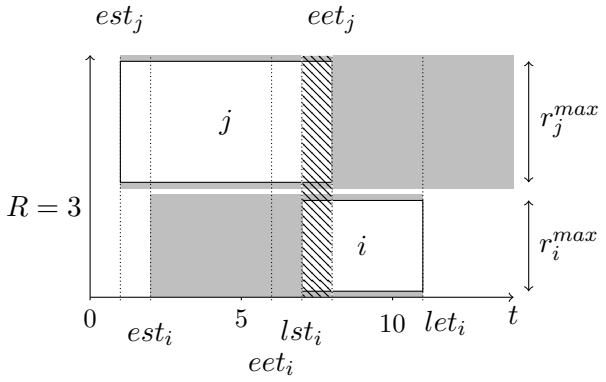


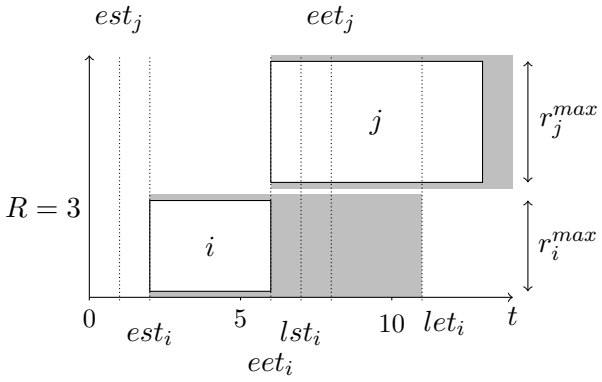

FIGURE 3.2 – Fonction $f_j(b_j(t))$.




(a) Si $i$ finit au temps 11...




(b) ...alors $i$ chevauche forcément l'activité $j$




(c) $est_j$ peut être ajusté

FIGURE 3.3 – Raisonnement disjonctif pour le CECSP.

Nous pouvons maintenant présenter l'adaptation du Time-Table disjonctif au cas du CECSP. Pour ce faire, nous commençons par adapter la définition d'intervalle minimum de superposition, puis nous présenterons les modifications apportées aux règles d'ajustement du CuSP afin que ces dernières soient applicables dans le cas du CECSP.

La notion d'intervalle minimum de superposition est légèrement plus difficile à définir que dans le cas du CuSP. En effet, cet intervalle représente l'ensemble minimal de points de temps tel qu'une activité $i$ est forcément en cours durant un de ces points et, comme au point $eet_i$, l'activité n'est pas

forcément en cours, il faut inclure le point "juste à gauche" de $eet_i$ dans $moi_i$. La définition peut donc s'écrire de la manière suivante.

**Définition 3.2.** *Soit $eet_i^-$ le point le plus proche de $eet_i > 0$, i.e. $\forall \delta > 0, |eet_i^- - eet_i| \leq \delta$. L'intervalle minimum de superposition d'une activité i,noté $moi_i$, est alors défini par $moi_i = [eet_i^-, lst_i]$ si i ne possède pas de partie obligatoire et $moi_i = \emptyset$ sinon.*

**Exemple 3.1.3.** *La figure 3.4 illustre l'intervalle minimum de superposition d'une activité i ne possédant pas de partie obligatoire. En effet, quelle que soit la position de l'activité i, elle intersecte forcément $moi_i$.*

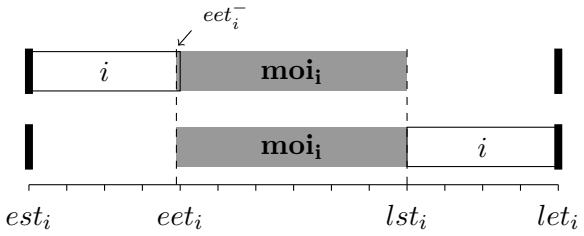



FIGURE 3.4 – Intervalle minimum de superposition d'une activité dans le cadre du CECSP.

Cependant, pour s'abstraire du point $eet_i^-$ et faciliter les notations, nous n'utiliserons pas cet intervalle pour définir les règles d'ajustement mais, à la place, nous les écrirons à l'aide d'inégalité stricte. Ces intervalles apparaîtront cependant sur les figures illustrant ces règles afin de faciliter leur compréhension et de les rapprocher de celles définies pour le CuSP.

Pour améliorer le raisonnement disjonctif, nous utilisons l'idée suivante : si deux activités $i$ et $j$ ne peuvent être exécutées en parallèle, alors la fenêtre de temps de $j$ ne peut contenir un point de temps que $i$ devra forcément chevaucher. Ceci nous permet de définir la règle d'ajustement suivante :

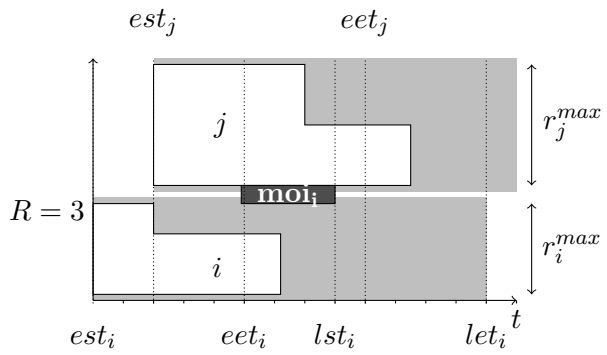
**Règle 3.2.** *Soient deux activités i et j telles que i ne possède pas de partie obligatoire et $r_i^{min} + r_j^{min} > R$. Si $est_j < eet_i$ et $eet_j > lst_i$ alors $est_j \geq eet_i$.*

**Exemple 3.1.4.** *Considérons l'instance à deux activités et avec une ressource de capacité 3 suivante :*

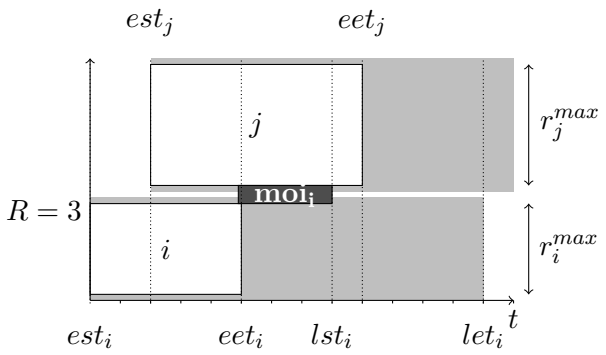
| $act$ | $est_\ell$ | $let_\ell$ | $W_\ell$ | $r_\ell^{min}$ | $r_\ell^{max}$ | $f_\ell(b_\ell(t))$ |
|---|---|---|---|---|---|---|
| $i$ | 0 | 13 | 35 | 2 | 3 | $2 * b_i(t) + 1$ |
| $j$ | 2 | 20 | 49 | 2 | 4 | $voir fig.$ 3.2 |

*La règle 3.2 est illustrée par la figure 3.5. Dans les figures 3.5a et 3.5b, on peut constater que si l'activité j commence à $est_j$, alors elle intersecterait complètement $moi_i$ et il serait impossible d'ordonnancer i. Sur la figure 3.5c, $est_j$ a été ajusté et l'activité j ne peut commencer avant $eet_i$.*

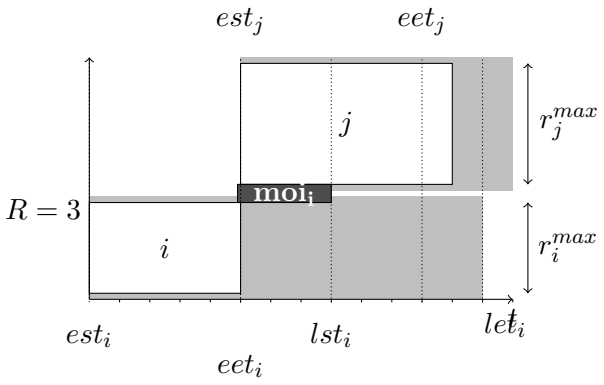
La règle 3.2 compare seulement les consommations de $i$ et de $j$ avec la capacité de la ressource $R$. Cependant, les consommations obligatoires des autres activités peuvent ne pas laisser $R$ unités de ressource durant l'intersection de $i$ et de $j$. La règle suivante prend donc en considération le profil obligatoire de la ressource.




(a) Si $j$ commence au temps 2...




(b) ...alors $j$ chevauche forcément $moi_i$




(c) $est_j$ peut être ajusté

FIGURE 3.5 – Raisonnement disjonctif restreint pour le CECSP.

**Règle 3.3.** *Soient $i$ et $j$ deux activités qui ne possèdent pas de partie obligatoire et telles que $r_i^{min} + r_j^{min} + \min_{eet_i \leq t \leq lst_i} TT_{\mathcal{A}}(t) > R$. Si $est_j < eet_i$ et $eet_j > lst_i$ alors $est_j \geq eet_i$.*

**Exemple 3.1.5.** *Considérons l'instance à deux activités et avec une ressource de capacité 3 suivante :*

| $act$ | $est_i$ | $let_i$ | $W_i$ | $r_i^{min}$ | $r_i^{max}$ | $f_i(b_i(t))$ |
|---|---|---|---|---|---|---|
| $i$ | 2 | 11 | 21 | 1 | 2 | $2 * b_i(t) + 1$ |
| $j$ | 1 | 20 | 14 | 1 | 2 | $b_i(t)$ |
| $k$ | 2 | 11 | 18 | 2 | 2 | $\frac{1}{3} * b_i(t) + \frac{4}{3}$ |

*Les activités 1 et 2 ne possèdent pas de partie obligatoire tandis que l'activité 3 est forcément en cours d'exécution durant l'intervalle $[2, 11[$.*

*L'intervalle $moi_i = [4, 8]$ est complètement inclus dans l'intervalle formé par $est_j$ et $eet_j$, i.e. $[1, 10[$. De plus, le minimum du profil de consommation de la ressource dans $moi_i$ est de 2. Donc, les activités $i$ et $j$, consommant au minimum 1 unité de ressource, ne peuvent se chevaucher dans l'intervalle $[4, 8]$. Donc $est_j$ peut être ajustée à 5.*

Certains raisonnements mis en place dans le cadre du CuSP peuvent donc être facilement adaptés au cas du CECSP en considérant des activités rectangulaires de durée ou de consommation minimale. Cependant, ces règles ne prenant en considération que des cas "extrêmes", elles demeurent moins

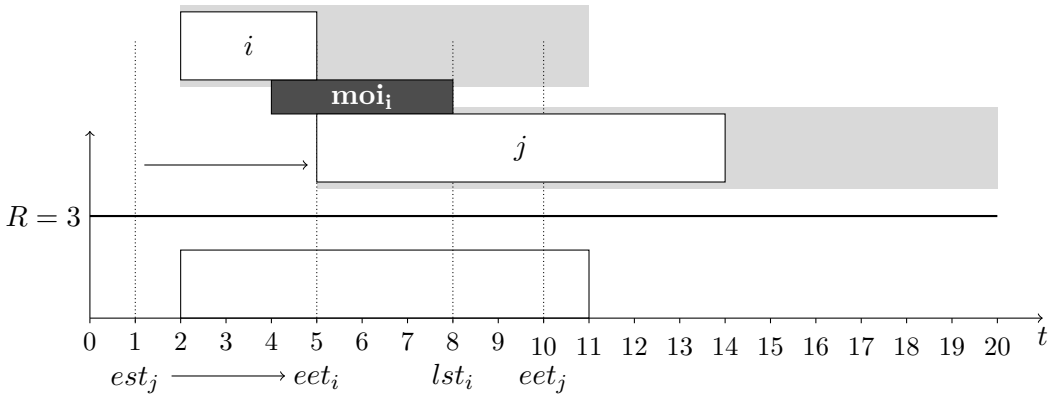



FIGURE 3.6 – Illustration du Time-Table disjonctif pour le CECSP.

efficaces que dans le cas du CuSP. C'est pourquoi, dans la section 3.2, nous avons choisi d'adapter un des raisonnements les plus forts pour le CuSP dans le cadre du CECSP : le raisonnement énergétique.

La prochaine sous-section présente quant à elle un raisonnement couplant modélisation de problème de flots et Time-Table.

### 3.1.3 Le Time-Table basé sur les flots

Le raisonnement décrit dans cette sous-section propose d'intégrer le raisonnement Time-Table dans un programme linéaire basé sur un problème de flots. Ce programme linéaire peut ensuite être utilisé comme algorithme de vérification, ou checker, pour s'assurer que les valeurs des domaines des variables ne vont pas amener à une incohérence. Il peut aussi être couplé avec des algorithmes de propagation tels que ceux décrits dans la sous-section précédente pour décrire un raisonnement complet.

L'idée de l'algorithme de vérification est donc de s'assurer qu'une fois les parties obligatoires des activités fixées, l'aire disponible est suffisante pour ordonnancer le reste des activités en tenant compte des fenêtres de temps de chaque activité. Pour ce faire, nous allons, dans un premier temps, contraindre chaque activité à consommer une quantité de ressource supérieure à $r_i^{min}$ dans l'intervalle $[lst_i, eet_i[$ et ensuite, nous relâchons la contrainte de consommation minimale en autorisant l'activité à consommer une quantité de ressource comprise entre 0 et $r_i^{max}$ durant le reste de l'intervalle $[est_i, let_i]$. Ceci peut être vu comme une variante de la relaxation préemptive du CECSP, décrite dans la section 1.2. Clairement, comme il s'agit d'une relaxation du CECSP, si aucune solution n'existe alors l'instance du CECSP correspondante ne possède pas de solutions.

Pour modéliser cette relaxation, nous introduisons $(t_q)_{q\in\mathcal{Q}}$, $|\mathcal{Q}| \leq 4 * n$, la suite croissante des bornes des domaines des variables, i.e. $(t_q)$ est composée des dates de début au plus tôt et au plus tard ainsi que des dates de fin correspondantes. Pour simplifier les notations, nous notons $\mathcal{Q}^*$ l'ensemble $\mathcal{Q} \setminus \max\{q \in \mathcal{Q}\}$. Nous définissons ensuite deux ensembles de variables. Le premier correspond à la quantité de ressource consommée par une activité $i$ dans la période $[t_q, t_{q+1}[$ et est noté $b_{iq},\ i \in \mathcal{A},\ q \in \mathcal{Q}^*$. Le second permet de modéliser la quantité d'énergie apportée à une activité dans cette même période et est noté $w_{iq},\ i \in \mathcal{A},\ q \in \mathcal{Q}^*$. Le problème de CECSP relâché peut alors se formuler par le

programme linéaire suivant :

$$\sum_{i \in \mathcal{A}} b_{iq} \leq R(t_{q+1} - t_q) \qquad \forall q \in \mathcal{Q}^* \tag{3.1}$$

$$b_{iq} \geq r_i^{min}(t_{q+1} - t_q) \qquad \forall i \in \mathcal{A},\ \forall q \in \mathcal{Q}^* \mid lst_i \leq t_q \leq eet_i \tag{3.2}$$

$$b_{iq} \leq r_i^{max}(t_{q+1} - t_q) \qquad \forall i \in \mathcal{A},\ \forall q \in \mathcal{Q}^* \tag{3.3}$$

$$b_{iq} = 0 \qquad \forall i \in \mathcal{A},\ \forall q \in \mathcal{Q}^* \mid t_q \notin [est_i, let_i] \tag{3.4}$$

$$w_{iq} \leq a_{ip} b_{iq} + c_{ip}(t_{q+1} - t_q) \qquad \forall i \in \mathcal{A},\ \forall q \in \mathcal{Q}^*,\ \forall p \in \mathcal{P}_i \tag{3.5}$$

$$w_{iq} \leq M b_{iq} \qquad \forall i \in \mathcal{A} \mid r_i^{min} \neq 0,\ \forall q \in \mathcal{Q}^* \tag{3.6}$$

$$\sum_{q \in \mathcal{Q}^*} w_{iq} = W_i \qquad \forall i \in \mathcal{A} \tag{3.7}$$

Dans cette formulation, la contrainte (3.1) permet d'assurer que la capacité de la ressource n'est pas excédée.

La contrainte (3.2) modélise la partie obligatoire d'une activité. En effet, la contrainte affirme que l'activité doit consommer une quantité de ressource supérieure à $r_i^{min}$ durant l'intervalle $[lst_i, eet_i[$.

La contrainte (3.3) garantit que l'activité ne consomme pas plus de $r_i^{max}$ de la ressource durant son exécution.

La contrainte (3.4) fixe la consommation de la ressource à zéro en dehors de l'intervalle $[est_i, let_i]$.

Les contraintes (3.5) et (3.6) permettent de modéliser l'énergie apportée à l'activité $i$ en fonction de la quantité de ressource consommée par cette même activité dans l'intervalle $[t_q, t_{q+1}]$. Notons d'abord l'utilisation d'une inégalité dans la contrainte (3.5) et non d'une égalité. La justification de la validité de cette contrainte est faite dans la section 5.1. La contrainte (3.6) stipule quant à elle que, pour une constante $M$ suffisament grande, si l'activité ne consomme pas de ressource, i.e. $b_{iq} = 0$, alors elle ne reçoit pas d'énergie, i.e. $w_{iq} = 0$. Notons que cette contrainte ne peut être définie dans le cas où $r_i^{min} = 0$ car une telle constante $M$ ne peut être trouvée. Dans ce cas-là, nous ommetons cette contrainte et ceci ne contredit pas la véracité de la formulation. Cependant, l'algorithme de vérification sera moins fort dans le cas où de nombreuses activités vérifient $r_i^{min} = 0$.

La contrainte (3.7) assure que la quantité d'énergie requise est bien apportée à l'activité.

Notons que le modèle possède $2 \times (n \times (4n - 1)) = 8n^2 - 2n$ variables continues et au plus $4n^2(4 + P) + n(1 - P) - 1$ contraintes, avec $P = \max_{i \in \mathcal{A}} |\mathcal{P}_i|$.

**Exemple 3.1.6.** *Considérons l'instance à 3 activités et avec une ressource de capacité $R = 3$ suivante :*

| *act.* | $est_i$ | $let_i$ | $W_i$ | $r_i^{min}$ | $r_i^{max}$ | $lst_i$ | $eet_i$ | $f_i(b_i(t))$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 2 | 4 | 2 | 2 | 0 | 2 | $b_i(t)$ |
| 2 | 4 | 6 | 4 | 2 | 2 | 4 | 6 | $b_i(t)$ |
| 3 | 0 | 6 | 10 | 1 | 2 | 1 | 5 | $b_i(t)$ |

*Considérons maintenant le programme linéaire défini ci-dessus associé à cette instance. La suite $t_q$ est formée des six éléments suivants : $t_0 = 0,\ t_1 = 1,\ t_2 = 2,\ t_3 = 4,\ t_4 = 5,\ t_5 = 6$. Notons que, comme*

*les fonctions de rendement correspondent à la fonction identité, la variable $w_{iq}$ n'a pas besoin d'être définie ici. Le programme linéaire est donc le suivant :*

$$b_{3q} + 2 \leq 3 \qquad q \neq 2 \qquad (3.1a)$$
$$b_{32} \leq 6 \qquad (3.1b)$$
$$b_{10} = b_{11} = 2 \qquad (3.2a)$$
$$b_{23} = b_{24} = 2 \qquad (3.2b)$$
$$b_{32} \geq 2 \qquad (3.2c)$$
$$b_{3q} \geq 1 \qquad q = 1, 3 \qquad (3.2d)$$
$$b_{3q} \leq 2 \qquad q = 0, 4 \qquad (3.3a)$$
$$b_{3q} \leq 2 \qquad q = 1, 3 \qquad (3.3b)$$
$$b_{32} \leq 4 \qquad (3.3c)$$
$$b_{1q} = 0 \qquad q = 2..4 \qquad (3.4a)$$
$$b_{2q} = 0 \qquad q = 0..2 \qquad (3.4b)$$
$$b_{30} + b_{31} + b_{32} + b_{33} + b_{34} = 10 \qquad (3.7a)$$

*Comme la contrainte (3.1a) borne $b_{3q}$ supérieurement par 1 pour $q = 0, 1, 3, 4$, la contrainte (3.7a) implique que $b_{32} = 6$ ce qui est en contradiction avec la contrainte (3.3c). Le programme linéaire détecte donc l'infaisabilité de l'instance. Notons aussi que le raisonnement Time-Table disjonctif n'aurait pas détecté l'incohérence.*

Dans cette section, nous avons montré que certaines extensions des relaxations du CuSP était directement adaptable au cas du CECSP. En effet, nous avons décrit l'extension des raisonnements Time-Table, disjonctif et Time-Table disjonctif. D'autres raisonnements existants font aussi partie de cette catégorie par exemple le Edge-Finding et ses extensions. Nous avons aussi montré que la programmation linéaire pouvait être utilisée pour décrire un algorithme de vérification qui pouvait être couplée à des algorithmes d'ajustements existants.

Dans la section suivante nous décrivons une extension non triviale du raisonnement énergétique, un des raisonnements les plus forts pour le CuSP.

## 3.2 Algorithme de filtrage du raisonnement énergétique

La section ci-dessous décrit l'adaptation du raisonnement énergétique, introduit dans [JE89] pour la contrainte cumulative, et décrit dans la sous-section 2.2.3. Nous commençons, dans un premier temps, par décrire l'algorithme de vérification de ce raisonnement, puis nous présenterons les règles d'ajustements qui peuvent être mises en place pour filtrer les domaines des variables. Enfin, la dernière partie de cette section sera consacrée à la caractérisation des intervalles d'intérêt pour l'algorithme de vérification et pour les règles d'ajustement.

### 3.2.1 Algorithme de vérification

#### Condition nécessaire d'existence de solution

Pour décrire l'algorithme de vérification, nous rappelons d'abord l'idée principale sur laquelle repose le raisonnement énergétique. Le principe est donc, étant donné un intervalle $[t_1, t_2[$, de calculer les consommations minimales de ressource des activités dans cet intervalle et de les comparer à la quantité de ressource disponible dans ce même intervalle. Si la ressource disponible n'est pas suffisante pour ordonnancer les consommations minimales de toutes les activités, une incohérence est détectée.

Dans le cas du CuSP, la quantité de ressource requise par une activité pouvait être calculée de manière directe. Ici, ce calcul sera fait en deux fois : nous calculons d'abord la quantité d'énergie requise par une activité à l'intérieur de l'intervalle $[t_1, t_2[$, notée $\underline{w}(i, t_1, t_2)$, puis nous traduisons cette énergie en une quantité de ressource, notée $\underline{b}(i, t_1, t_2)$. Ceci nous permettra ensuite de la comparer avec la ressource disponible dans $[t_1, t_2[$.

Formellement, ces quantités sont représentées par les expressions suivantes :

$$\underline{w}(i, t_1, t_2) = \min \int_{t_1}^{t_2} f_i(b_i(t))dt \qquad \text{sous (A.1)-(A.4)} \tag{3.8}$$

$$\underline{b}(i, t_1, t_2) = \min \int_{t_1}^{t_2} b_i(t)dt \qquad \text{sous (A.1)-(A.4)} \tag{3.9}$$

Comme pour le cas du CuSP, la fonction de marge, notée $SL(t_1, t_2)$, permet de mesurer l'écart entre la quantité de ressource disponible et les consommations minimales de toutes les tâches dans l'intervalle $[t_1, t_2]$. Cette fonction est définie de la manière suivante :

$$SL(t_1, t_2) = R(t_2 - t_1) - \sum_{i \in A} \underline{b}(i, t_1, t_2)$$

Ceci nous permet d'énoncer la condition nécessaire d'existence d'une solution qui est à la base de l'algorithme de vérification du raisonnement énergétique :

**Théorème 3.1.** *Soit $\mathcal{I}$ une instance du CECSP. S'il existe $t_1 < t_2 \in \mathbb{R}^2$ tel que $SL(t_1, t_2) < 0$ alors $\mathcal{I}$ ne peut pas avoir de solution.*

*Démonstration.* Par l'absurde, supposons qu'il existe $t_1 < t_2 \in \mathbb{R}^2$ tel que $SL(t_1, t_2) < 0$ et que l'instance $\mathcal{I}$ soit satisfiable. Par définition, $\underline{b}(i, t_1, t_2)$ est la quantité de ressource minimale que doit consommer l'activité $i$ dans l'intervalle $[t_1, t_2[$.

Donc, dans toute solution réalisable, nous avons :

$$\int_{t_1}^{t_2} b_i(t)dt \geq \underline{b}(i, t_1, t_2)$$

$$\Rightarrow \sum_{i \in \mathcal{A}} \int_{t_1}^{t_2} b_i(t)dt \geq \sum_{i \in \mathcal{A}} \underline{b}(i, t_1, t_2) > R(t_2 - t_1)$$

Et ceci contredit le fait que $\sum_{i \in \mathcal{A}} b_i(t) \leq R(t_2 - t_1)$. □

Dans un premier temps, nous allons nous intéresser au calcul de $\underline{w}(i, t_1, t_2)$, le calcul de $\underline{b}(i, t_1, t_2)$ sera détaillé dans un second temps.

### Énergie minimale dans un intervalle

Pour calculer $\underline{w}(i, t_1, t_2)$, nous analysons les différentes configurations de la consommation minimale d'une activité. Remarquons que les configurations conduisant à une consommation minimale dans l'intervalle $[t_1, t_2[$ sont celles où l'activité est ordonnancée à $r_i^{max}$ à l'extérieur de cet intervalle. Ces configurations, décrites dans la figure 3.7, peuvent être regroupées en trois catégories :

- l'activité est *calée à gauche* (figure 3.7a, 3.7d et 3.7g) : l'activité démarre à $est_i$ et est ordonnancée à $r_i^{max}$ pendant l'intervalle $[est_i, t_1[$ ;
- l'activité est *calée à droite* (figure 3.7c, 3.7f et 3.7i) : l'activité finit à $let_i$ et est ordonnancée à $r_i^{max}$ pendant l'intervalle $[t_2, let_i[$ ;
- l'activité est *centrée* (figure 3.7b, 3.7e et 3.7h) : l'activité occupe tout l'intervalle $[t_1, t_2[$, soit en étant ordonnancée à $r_i^{max}$ pendant l'intervalle $[est_i, t_1[ \cup [t_2, let_i[$, soit en étant ordonnancée à $r_i^{min}$ durant tout l'intervalle $[t_1, t_2[$.
  En effet, lorsque l'activité est ordonnancée à $r_i^{max}$ pendant l'intervalle $[est_i, t_1[ \cup [t_2, let_i[$, il peut arriver que la quantité d'énergie restant à apporter à l'activité durant l'intervalle $[t_1, t_2[$ ne soit pas suffisante pour assurer la satisfaction de la contrainte de consommation minimale (1.7). Le cas où l'activité est ordonnancée à $r_i^{min}$ durant tout l'intervalle $[t_1, t_2[$ doit donc être considéré.

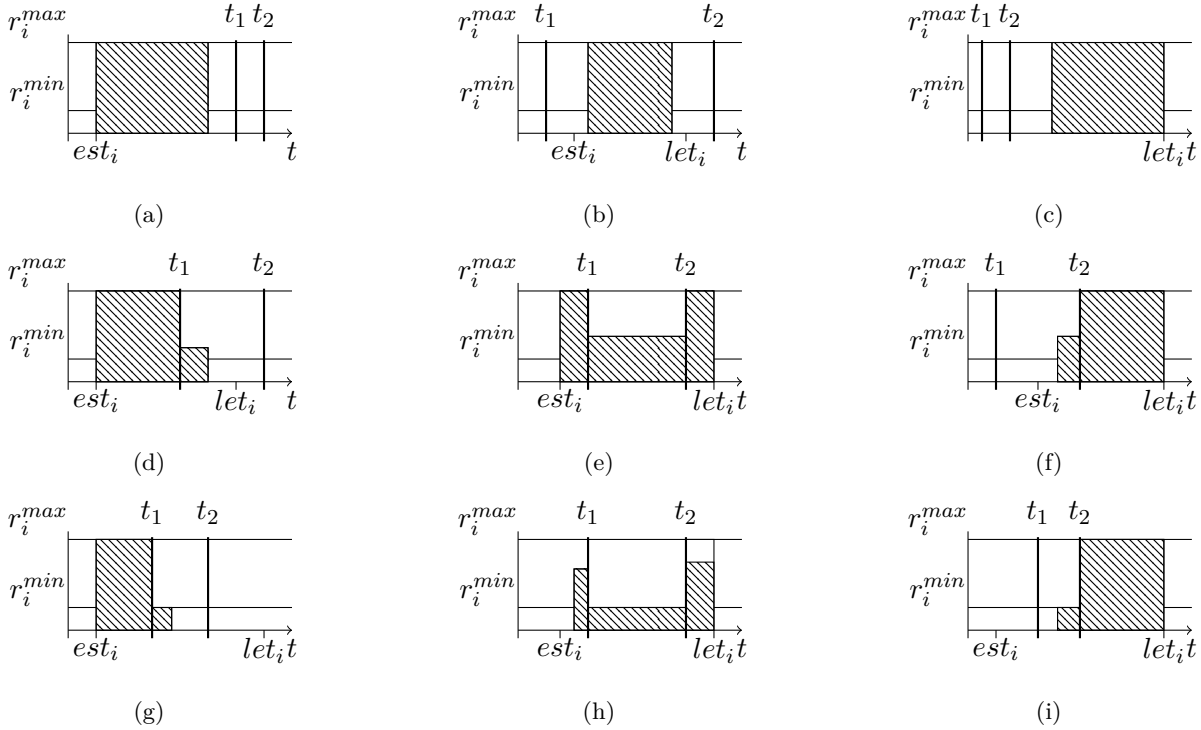



FIGURE 3.7 – Les différentes configurations menant à une consommation minimale à l'intérieur de $[t_1, t_2[$ pour le CECSP.

Il est facile de calculer l'expression de la consommation minimale d'énergie dans un intervalle pour une fonction $f_i$ croissante. En effet, les différentes configurations possibles étant toujours celles où l'activité est exécutée à son rendement maximum en dehors de l'intervalle $[t_1, t_2[$, il suffit de retrancher

à $W_i$ l'énergie produite par l'exécution de l'activité à $r_i^{max}$ en dehors de $[t_1, t_2[$. Il existe une exception à cette règle, produite par la contrainte de consommation minimale, mais ce cas est facilement traité puisque l'énergie minimale correspond alors à la configuration où la tâche est exécutée à son rendement minimal durant $[t_1, t_2[$.

Pour donner l'expression mathématique de $\underline{w}(i, t_1, t_2)$, nous introduisons trois notations. $\underline{w}_{LS}(i, t_1, t_2)$ (respectivement $\underline{w}_{RS}(i, t_1, t_2)$ et $\underline{w}_{CS}(i, t_1, t_2)$) correspond à la quantité d'énergie apportée à l'activité $i$ dans l'intervalle $[t_1, t_2[$ quand l'activité est calée à gauche (respectivement calée à droite et centrée). Formellement, ces trois quantités peuvent être exprimées de la manière suivante :

$$\underline{w}_{LS}(i, t_1, t_2) = \max\left(0\,,\, W_i - f_i(r_i^{max}) \max(0, t_1 - est_i)\right) \tag{3.10}$$

$$\underline{w}_{RS}(i, t_1, t_2) = \max\left(0\,,\, W_i - f_i(r_i^{max}) \max(0, let_i - t_2)\right) \tag{3.11}$$

$$\underline{w}_{CS}(i, t_1, t_2) = \max\left(f_i(r_i^{min}) * (t_2 - t_1)\,,\, W_i - f_i(r_i^{max})|[est_i, let_i] \setminus [t_1, t_2[|\right) \tag{3.12}$$

Alors, l'expression de l'énergie minimale est le minimum de ces trois quantités, i.e.

$$\underline{w}(i, t_1, t_2) = \min\left(\,\underline{w}_{LS}(i, t_1, t_2)\,,\, \underline{w}_{RS}(i, t_1, t_2)\,,\, \underline{w}_{CS}(i, t_1, t_2)\,\right) \tag{3.13}$$

Nous allons maintenant utiliser l'expression de $\underline{w}(i, t_1, t_2)$ et la fonction $f_i$ pour calculer $\underline{b}(i, t_1, t_2)$.

**Consommation minimale de la ressource**

Pour calculer l'expression de $\underline{b}(i, t_1, t_2)$, nous allons utiliser les propriétés de la fonction $f_i$. En effet, une fois que nous avons calculé $\underline{w}(i, t_1, t_2)$, nous voulons savoir quelle est la quantité minimale de ressource que nous devons fournir à la tâche pour obtenir cette quantité d'énergie dans l'intervalle $[t_1, t_2[$. Dans le cas où la fonction $f_i$ est l'identité, nous avons que : $\underline{b}(i, t_1, t_2) = \underline{w}(i, t_1, t_2)$. Dans les deux autres cas, i.e. $f_i$ est affine et $f_i$ est concave et affine par morceaux, soit $I = [t_1, t_2[ \cap [est_i, let_i[$, alors trouver $\underline{b}(i, t_1, t_2)$ revient à résoudre le programme suivant :

$$\text{minimiser} \int_I b_i(t)dt \tag{3.14}$$

$$\text{sous} \int_I f_i(b_i(t))dt \geq \underline{w}(i, t_1, t_2) \tag{3.15}$$

$$r_i^{min} \leq b_i(t) \leq r_i^{max} \tag{3.16}$$

En effet, l'objectif de ce programme est de minimiser la quantité de ressource consommée dans l'intervalle $[t_1, t_2[$ (équation (3.14)), tout en s'assurant que l'énergie requise, i.e. $\underline{w}(i, t_1, t_2)$, est bien apportée à l'activité (équation (3.15)).

Nous utilisons ensuite le lemme 1.1 pour simplifier ce programme. En effet, le lemme affirme que, si $f_i$ est affine ou concave et affine par morceaux, alors si une solution optimale $b_i(t)$ pour le programme ci-dessus existe, cette solution peut être transformée en une autre solution optimale vérifiant la propriété que $b_i(t)$ soit constante et inférieure ou égale à $\frac{\int_I b_i(t)dt}{|I|}$. Nous noterons $b_i$ cette constante. Le programme simplifié s'écrit de la manière suivante :

$$\text{minimiser } b_i|I| \tag{3.17}$$

$$\text{sous } f_i(b_i)|I| \geq \underline{w}(i,t_1,t_2) \tag{3.18}$$

$$r_i^{min} \leq b_i \leq r_i^{max} \tag{3.19}$$

Ce programme peut être réécrit de la façon suivante :

$$b_i = \min\left(b \in [r_i^{min}, r_i^{max}] \mid f_i(b) \geq \frac{\underline{w}(i,t_1,t_2)}{|I|}\right)$$

Nous pouvons remarquer que, si l'on suppose que $f_i(r_i^{max})(let_i - est_i) \geq W_i$, nous sommes sûrs que la solution optimale vérifie $b_i \leq r_i^{max}$. En effet, ceci est dû au fait qu'exécuter l'activité à une faible consommation de ressource a un meilleur rendement que son exécution à un rendement plus élevé, i.e. la fonction $\frac{f_i(b_i(t))}{b_i(t)}$ est décroissante.

De ce fait, seule la borne inférieure sur $b_i$, $r_i^{min}$, doit être considérée. Or, pour apporter l'énergie $\underline{w}(i,t_1,t_2)$ à l'activité dans l'intervalle $I$, sa consommation $b_i$ doit vérifier $f_i(b_i) \geq \frac{\underline{w}(i,t_1,t_2)}{|I|}$ et donc $b_i \geq f_i^{-1}\left(\frac{\underline{w}(i,t_1,t_2)}{|I|}\right)$. En couplant cette contrainte avec la contrainte de consommation minimale, nous obtenons, si $r_i^{min} \neq 0$ :

$$b_i = \max\left(r_i^{min}, f_i^{-1}\left(\frac{\underline{w}(i,t_1,t_2)}{|I|}\right)\right)$$

et si $r_i^{min} = 0$, nous avons :

$$b_i = f_i^{-1}\left(\frac{\underline{w}(i,t_1,t_2)}{|I|}\right)$$

Nous allons maintenant donner l'expression de $\underline{b}(i,t_1,t_2)$ en fonction des coefficients $a_{ip}$ et $c_{ip}$ de la fonction $f_i$. Pour simplifier la compréhension, nous commençons par détailler cette expression dans le cas où $f_i$ est affine puis nous la généraliserons dans le cas où $f_i$ est concave et affine par morceaux.

**Fonctions affines** Dans ce paragraphe, nous allons décrire l'expression de $\underline{b}(i,t_1,t_2)$ dans le cas où la fonction $f_i$ est affine, i.e. de la forme $a_i b_i(t) + c_i$. Dans ce cas-là, nous avons :

$$f_i^{-1}\left(\frac{\underline{w}(i,t_1,t_2)}{|I|}\right) = \frac{\underline{w}(i,t_1,t_2) - c_i|I|}{a_i|I|}$$

et donc, si $r_i^{min} \neq 0$ :

$$\underline{b}(i,t_1,t_2) = \max\left(r_i^{min}\frac{\underline{w}(i,t_1,t_2)}{f_i(r_i^{min})}, \frac{1}{a_i}\left(\underline{w}(i,t_1,t_2) - |I|c_i\right)\right) \tag{3.20}$$

et si $r_i^{min} = 0$ :

$$\underline{b}(i,t_1,t_2) = \frac{1}{a_i}\left(\underline{w}(i,t_1,t_2) - |I|c_i\right)$$

Notons que le premier cas correspond au cas où l'intervalle $I$ est suffisamment grand pour ordonnancer l'activité $i$ à $r_i^{min}$ et lui apporter l'énergie requise, i.e. $\underline{w}(i,t_1,t_2)$. En effet, comme ordonnancer $i$ à $r_i^{min}$ a le meilleur rendement, si cela est possible, on exécutera $i$ à $b_i = r_i^{min}$. Si, au contraire, l'intervalle n'est pas suffisamment grand, c'est $f_i^{-1}(\underline{w}(i,t_1,t_2)/|I|)$ que l'on va choisir. Ceci est illustré dans l'exemple 3.2.1.

**Exemple 3.2.1.** *Considérons les deux activités suivantes (voir figure 3.8) :*

| $act.$ | $est_i$ | $let_i$ | $W_i$ | $r_i^{min}$ | $r_i^{max}$ | $f_i(b_i(t))$ |
|---|---|---|---|---|---|---|
| $i$ | 2 | 8 | 21 | 2 | 5 | $b_i(t)+3$ |
| $j$ | 0 | 8 | 22 | 2 | 5 | $\frac{1}{2} * b_i(t) + \frac{1}{2}$ |

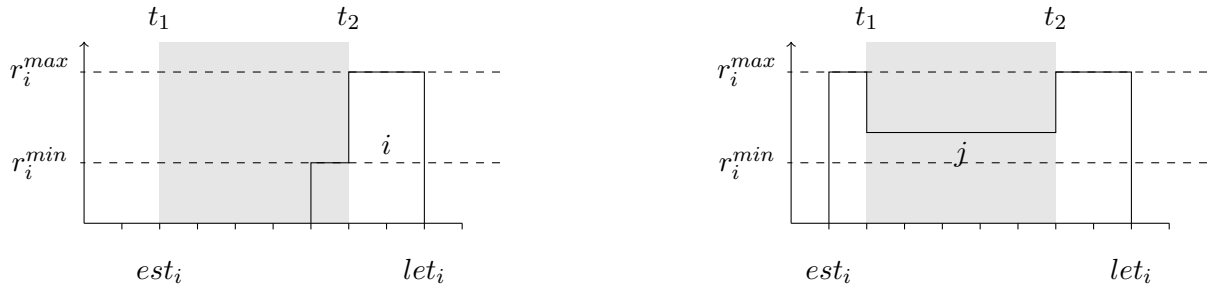
*Si nous calculons* $\underline{w}(i,t_1,t_2)$ *et* $\underline{b}(i,t_1,t_2)$ *sur l'intervalle* $[1,6[$*, nous obtenons :*

— $\underline{w}(i,1,6) = \underline{w}_{RS}(i,1,6) = 5$ *et*

— $\underline{b}(i,1,6) = \max(r_i^{min}\frac{\underline{w}(i,1,6)}{f_i(r_i^{min})}, \frac{1}{a_i}(\underline{w}(i,1,6) - c_i|I|) = \max(2\frac{5}{5}, \frac{1}{1}(5-3\times 4) = \max(2,-7) = 2$

— $\underline{w}(j,1,6) = \underline{w}_{CS}(j,1,6) = 10$ *et*

— $\underline{b}(j,1,6) = \max(r_i^{min}\frac{\underline{w}(j,1,6)}{f_i(r_i^{min})}, \frac{1}{a_i}(\underline{w}(j,1,6) - c_i|I|) = \max(2\frac{10}{\frac{3}{2}}, 2\times(10-\frac{1}{2}\times 5) = \max(\frac{40}{3}, 15) = 15$




(a) Activité $i$ (b) Activité $j$

FIGURE 3.8 – Consommation de ressource dans $[t_1,t_2[$ pour le CECSP avec fonction de rendement affine.

Ce raisonnement peut être étendu dans le cas où $f_i$ est concave et affine par morceaux.

**Fonctions concave et affine par morceaux** Dans le cas des fonctions concaves et affines par morceaux, le raisonnement décrit au paragraphe précédent peut être généralisé. En effet, comme dans le cas des fonctions affines, la fonction de rendement, $f_i(b_i(t))/b_i(t)$, est décroissante. Donc, il est toujours préférable d'exécuter l'activité avec une consommation de ressource aussi basse que possible. La condition selon laquelle on peut ou non exécuter l'activité $i$ avec une consommation $b_i$ dépend donc de la taille de l'intervalle $I$.

Si $r_i^{min} \neq 0$ et si l'intervalle est suffisamment grand, on va donc exécuter l'activité à $r_i^{min}$. Sinon, et si l'intervalle est suffisamment grand, nous essayons d'exécuter l'activité avec une consommation $b_i \in ]r_i^{min}, x_2^i]$, etc. Nous rappelons que les points $x_\ell^i$ correspondent aux points de rupture de la fonction $f_i$. Dans le cas où $r_i^{min} \neq 0$, l'expression de $\underline{b}(i,t_1,t_2)$ est formalisée ci-dessous :

$$f_i^{-1}\left(\frac{\underline{w}(i,t_1,t_2)}{|I|}\right) = \begin{cases} r_i^{min} & \text{si } |I| \geq \dfrac{\underline{w}(i,t_1,t_2)}{f_i(r_i^{min})} \\ \dfrac{\underline{w}(i,t_1,t_2) - c_{i1}|I|}{a_{i1}|I|} & \text{si } \dfrac{\underline{w}(i,t_1,t_2)}{f_i(r_i^{min})} > |I| > \dfrac{\underline{w}(i,t_1,t_2)}{f_i(x_2^i)} \\ \dfrac{\underline{w}(i,t_1,t_2) - c_{i2}|I|}{a_{i2}|I|} & \text{si } \dfrac{\underline{w}(i,t_1,t_2)}{f_i(x_2^i)} \geq |I| > \dfrac{\underline{w}(i,t_1,t_2)}{f_i(x_3^i)} \\ \vdots & \\ \dfrac{\underline{w}(i,t_1,t_2) - c_{iP_i}|I|}{a_{iP_i}|I|} & \text{si } \dfrac{\underline{w}(i,t_1,t_2)}{f_i(x_{P_i}^i)} \geq |I| > \dfrac{\underline{w}(i,t_1,t_2)}{f_i(r_i^{max})} \end{cases}$$

Et ceci nous donne l'expression de $\underline{b}(i,t_1,t_2)$ suivante :

$$\underline{b}(i,t_1,t_2) = \max\left\{ b_i^{min}\frac{\underline{w}(i,t_1,t_2)}{f_i(b_i^{min})}, \max_{p\in\mathcal{P}_i}\left(\frac{1}{a_{ip}}(\underline{w}(i,t_1,t_2) - |I|c_{ip})\right)\right\} \tag{3.21}$$

Dans le cas où $r_i^{min} = 0$, nous avons :

$$\underline{b}(i,t_1,t_2) = \max_{p\in\mathcal{P}_i}\left(\frac{1}{a_{ip}}(\underline{w}(i,t_1,t_2) - |I|c_{ip})\right)$$

Un exemple du calcul de $\underline{b}(i,t_1,t_2)$ dans le cas d'une fonction $f_i$ concave et linéaire par morceaux est décrit dans l'exemple 3.2.2.

**Exemple 3.2.2.** *Considérons l'activité suivante (voir figure 3.9) :*

| *act.* | $est_i$ | $let_i$ | $W_i$ | $r_i^{min}$ | $r_i^{max}$ | $f_i(b)$ |
|---|---|---|---|---|---|---|
| $i$ | 2 | 8 | 70 | 2 | 5 | $3b+1$ *si* $b\in[2,3]$ |
| | | | | | | $2b+4$ *si* $b\in]3,4]$ |
| | | | | | | $b+8$ *si* $b\in]4,5]$ |

*Si nous calculons $\underline{w}(i,t_1,t_2)$ et $\underline{b}(i,t_1,t_2)$ sur l'intervalle $[2,6[$, nous obtenons :*

— $\underline{w}(i,2,6) = \underline{w}_{RS}(i,2,6) = 44$ *et*

— $\underline{b}(i,2,6) = \max(r_i^{min}\frac{\underline{w}(i,2,6)}{f_i(r_i^{min})}, \max_{p\in\{1,2,3\}}\frac{1}{a_{ip}}(\underline{w}(i,2,6) - c_{ip}|I|))$

$$= \max\left(2\frac{44}{7}, \max\left\{\frac{1}{3}(44-4), \frac{1}{2}(44-4\times4), \frac{1}{1}(44-8\times4)\right\}\right)$$
$$= \max(\frac{88}{7}, \max\{\frac{40}{3}, 14, 12\})$$
$$= 14$$

*Dans ce cas, c'est la seconde partie de la fonction $f_i$ qui est utilisée, i.e. $f_i(b) = 2b+4$ si $b\in]3,4]$.*

*Si maintenant nous calculons $\underline{w}(i,t_1,t_2)$ et $\underline{b}(i,t_1,t_2)$ sur l'intervalle $[4,6[$, nous obtenons :*

— $\underline{w}(i,4,6) = \underline{w}_{CS}(j,4,6) = 18$ *et*

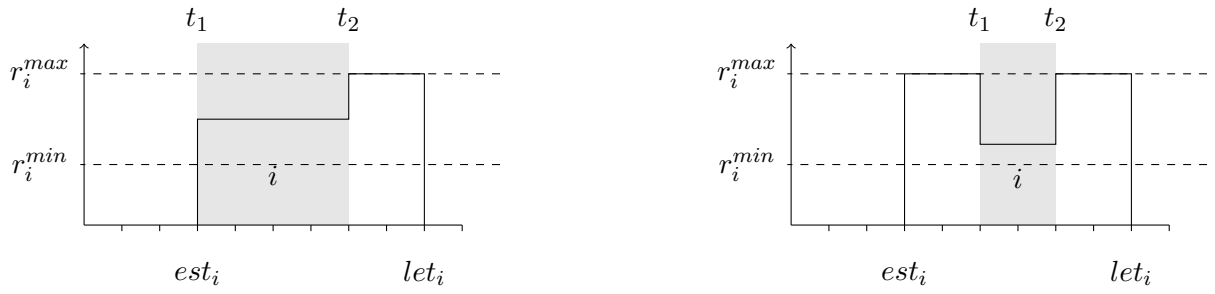
— $\underline{b}(i,4,6) = \max(r_i^{min}\frac{\underline{w}(i,4,6)}{f_i(r_i^{min})}, \max_{p\in\{1,2,3\}}\frac{1}{a_{ip}}(\underline{w}(i,4,6) - c_{ip}|I|))$

$$= \max\left(2\frac{18}{7}, \max\left\{\frac{1}{3}(18-2), \frac{1}{2}(18-4\times2), \frac{1}{1}(18-8\times2)\right\}\right)$$
$$= \max(\frac{36}{7}, \max\{\frac{16}{3}, 5, 2\})$$
$$= \frac{16}{3}$$

*Et dans ce cas, c'est la première partie de la fonction $f_i$ qui est utilisée, i.e. $f_i(b) = 3b+1$ si $b\in]2,3]$.*

### 3.2.2 Les ajustements de bornes

Dans cette sous-section, nous décrivons les ajustements qui peuvent être faits sur les fenêtres de temps des activités. Tout d'abord, nous introduisons les notations suivantes : $\underline{b}_{LS}(i,t_1,t_2)$ (respectivement $\underline{b}_{RS}(i,t_1,t_2)$ et $\underline{b}_{CS}(i,t_1,t_2)$) correspond à la quantité de ressource consommée par l'activité $i$




(a) Consommation minimale sur $[2, 6[$

(b) Consommation minimale sur $[4, 6[$

FIGURE 3.9 – Consommation de ressource dans $[t_1, t_2[$ pour le CECSP avec fonction de rendement concave et affine par morceaux.

dans l'intervalle $[t_1, t_2[$ quand l'activité est calée à gauche (respectivement calée à droite et centrée). Formellement, ces trois quantités peuvent être exprimées de la manière suivante :

$$\underline{b}_{LS}(i, t_1, t_2) = \max\left\{ b_i^{min} \frac{\underline{w}_{LS}(i, t_1, t_2)}{f_i(b_i^{min})}, \max_{p \in \mathcal{P}_i} \left( \frac{1}{a_{ip}} (\underline{w}_{LS}(i, t_1, t_2) - |I| c_{ip}) \right) \right\} \tag{3.22}$$

$$\underline{b}_{RS}(i, t_1, t_2) = \max\left\{ b_i^{min} \frac{\underline{w}_{RS}(i, t_1, t_2)}{f_i(b_i^{min})}, \max_{p \in \mathcal{P}_i} \left( \frac{1}{a_{ip}} (\underline{w}_{RS}(i, t_1, t_2) - |I| c_{ip}) \right) \right\} \tag{3.23}$$

$$\underline{b}_{CS}(i, t_1, t_2) = \max\left\{ b_i^{min} \frac{\underline{w}_{CS}(i, t_1, t_2)}{f_i(b_i^{min})}, \max_{p \in \mathcal{P}_i} \left( \frac{1}{a_{ip}} (\underline{w}_{CS}(i, t_1, t_2) - |I| c_{ip}) \right) \right\} \tag{3.24}$$

Nous allons décrire deux règles d'ajustements pour les fenêtres de temps du CECSP. La première permet d'ajuster $lst_i$ et la seconde $est_i$ ;nous avons des ajustements symétriques pour $eet_i$ et $let_i$.

Pour les premiers ajustements décrits, nous allons essayer de faire démarrer l'activité après $t_1$ et si la quantité de ressource disponible n'est pas suffisante pour ordonnancer les consommations minimales de toutes les activités -– excepté celle de l'activité $i$ qui est remplacée par $\underline{b}_{RS}(i, t_1, t_2)$ — alors, nous pouvons déduire que l'activité doit commencer avant $t_1$.

**Règle 3.4.** *S'il existe un intervalle $[t_1, t_2[$ avec $t_1 > est_i$ et une activité $i$ pour lesquels :*

$$\sum_{\substack{j \in \mathcal{A} \\ j \neq i}} \underline{b}(j, t_1, t_2) + \underline{b}_{RS}(i, t_1, t_2) > R(t_2 - t_1)$$

*alors, on a :*

$$lst_i \leq t_1 - \frac{1}{r_i^{max}} \left( \sum_{\substack{j \in \mathcal{A} \\ j \neq i}} \underline{b}(j, t_1, t_2) + \underline{b}_{RS}(i, t_1, t_2) - R(t_2 - t_1) \right)$$

*Démonstration.* Soient $i$ et $[t_1, t_2[$ vérifiant la condition de la règle 3.4. Supposons que l'activité $i$ démarre après $t_1$.

La consommation minimale de $i$ dans $[t_1, t_2[$ est donc $\underline{b}_{RS}(i, t_1, t_2)$. On a donc que $\sum_{\substack{j \in \mathcal{A} \\ j \neq i}} \underline{b}(j, t_1, t_2) + \underline{b}_{RS}(i, t_1, t_2)$ est la consommation minimale de toutes les activités dans $[t_1, t_2[$ quand l'activité $i$ commence après $t_1$. Donc, si cette quantité est plus grande que la quantité de ressource disponible dans

l'intervalle $[t_1, t_2[$, i.e. $R(t_2 - t_1)$, on obtient une contradiction avec le fait que l'instance soit réalisable et $i$ doit commencer avant $t_1$.

Pour calculer la nouvelle date de début au plus tard de $i$, remarquons que la quantité de ressource devant être consommée avant $t_1$ est d'au moins $\sum_{\substack{j \in \mathcal{A} \\ j \neq i}} \underline{b}(j, t_1, t_2) + \underline{b}_{RS}(i, t_1, t_2) - R(t_2 - t_1)$, i.e. ce qui ne "rentrait" pas dans $[t_1, t_2[$. L'objectif étant d'obtenir une borne supérieure sur la date de début de $i$, nous cherchons donc à exécuter cette partie de l'activité le plus rapidement possible.

Le temps minimal requis pour réaliser cette consommation étant obtenu en exécutant l'activité à $r_i^{max}$, nous obtenons comme borne supérieure sur la date de début de $i$ : $t_1 - 1/r_i^{max} \times \left( \sum_{\substack{j \in \mathcal{A} \\ j \neq i}} \underline{b}(j, t_1, t_2) + \underline{b}_{RS}(i, t_1, t_2) - R(t_2 - t_1) \right)$ □

De manière similaire, nous avons les ajustements suivants sur la date de début au plus tôt d'une activité.

**Règle 3.5.** *S'il existe un intervalle* $[t_1, t_2[$ *avec* $t_2 > let_i$ *et une activité* $i$ *telle que* $r_i^{min} \neq 0$ *pour lesquels :*

$$\sum_{\substack{j \in \mathcal{A} \\ j \neq i}} \underline{b}(j, t_1, t_2) + \min(\underline{b}_{CS}(i, t_1, t_2), \underline{b}_{LS}(i, t_1, t_2)) > R(t_2 - t_1)$$

*alors*

$$est_i \geq t_2 - \frac{1}{r_i^{min}} \left( R(t_2 - t_1) - \sum_{\substack{j \in \mathcal{A} \\ j \neq i}} \underline{b}(j, t_1, t_2) \right)$$

*Démonstration.* Soient $i$ et $[t_1, t_2[$ vérifiant la condition de la règle 3.4. Nous allons décider si $i$ peut commencer avant $t_1$. Les configurations où l'activité $i$ démarre avant $t_1$ et consomme le moins de ressource possible dans $[t_1, t_2[$ sont les configurations où $i$ est soit calée à gauche, soit centrée.

La consommation minimale de $i$ dans l'intervalle $[t_1, t_2[$ quand $i$ commence avant $t_1$ est donc $\min(\underline{b}_{LS}(i, t_1, t_2), \underline{b}_{CS}(i, t_1, t_2))$. On a donc que $\sum_{\substack{j \in \mathcal{A} \\ j \neq i}} \underline{b}(j, t_1, t_2) + \min(\underline{b}_{LS}(i, t_1, t_2), \underline{b}_{CS}(i, t_1, t_2))$ est la consommation minimale de toutes les activités dans $[t_1, t_2[$ quand l'activité $i$ commence avant $t_1$. Donc, si cette quantité est plus grande que la quantité de ressource disponible dans l'intervalle $[t_1, t_2[$, i.e. $R(t_2 - t_1)$, on obtient une contradiction avec le fait que l'instance soit réalisable et $i$ doit commencer après $t_1$.

Pour calculer la nouvelle date de début au plus tard de $i$, remarquons que la quantité de ressource disponible dans $[t_1, t_2[$ pour exécuter $i$ est de $R(t_2 - t_1) - \sum_{\substack{j \in \mathcal{A} \\ j \neq i}} \underline{b}(j, t_1, t_2)$. L'objectif étant d'obtenir une borne inférieure sur la date de début de $i$, nous cherchons donc à ordonnancer cette partie de l'activité le moins rapidement possible.

Le temps maximal requis pour réaliser cette consommation étant obtenu en exécutant l'activité à $r_i^{min}$, et, comme $r_i^{min} \neq 0$, l'activité ne peut être interrompue, nous savons que l'activité doit être en cours à $t_2$. Nous obtenons alors comme borne inférieure sur la date de début de $i$ : $t_2 - 1/r_i^{min} \left( R(t_2 - t_1) - \sum_{\substack{j \in \mathcal{A} \\ j \neq i}} \underline{b}(j, t_1, t_2) \right)$ □

**Exemple 3.2.3.** *Considérons l'instance à* 3 *activités et avec* $R = 5$ *suivante :*

| $act.$ | $est_i$ | $let_i$ | $W_i$ | $r_i^{min}$ | $r_i^{max}$ | $f_i(b_i(t))$ |
|---|---|---|---|---|---|---|
| 1 | 0 | 6 | 28 | 1 | 5 | $2b_i(t)+1$ |
| 2 | 2 | 6 | 32 | 2 | 5 | $b_i(t)+5$ |
| 3 | 2 | 5 | 6 | 2 | 2 | $b_i(t)$ |

*Une solution réalisable est décrite par la figure 3.10.*

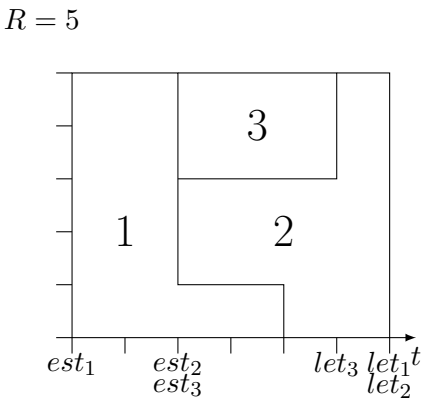



FIGURE 3.10 – Une solution réalisable du CECSP.

*Nous allons ajuster la fenêtre de temps de l'activité* 1. *Pour cela, considérons l'intervalle* $[t_1, t_2[ = [2, 5[$. *On a :*

— $\underline{b}(2, 2, 5) = 7$

— $\underline{b}(3, 2, 5) = 6$

— $\underline{b}_{RS}(1, 2, 5) = 7$ *et* $\underline{b}_{CS}(1, 2, 5) = 3$

*Nous avons donc :* $\sum_{j \in A;\ j \neq i} \underline{b}(j, 2, 5) + \underline{b}_{RS}(1, 2, 5) = 7 + 7 + 6 = 20 > 5(5 - 2) = 15$. *Donc* $lst_1$ *peut être ajustée à* $2 - \frac{1}{5}(20 - 15) = 1$ . *En effet, la quantité de ressource disponible dans l'intervalle* $[2, 5[$ *pour ordonnancer l'activité* 1 *est de* $15-7-6 = 2$. *Donc, si l'activité* 1 *commence après* $t_1$, *elle ne peut consommer que* 2 *unités de ressource dans* $[2, 5[$. *Or, il faudrait* $\underline{b}_{RS}(1, 2, 5) = 7$ *unités de ressource disponibles dans* $[2, 5[$ *pour que* 1 *puisse démarrer avant* $t_1$. *Donc l'activité* 1 *ne peut commencer après* $t_1$ *et, au moins* $5 = 7 - 2$ *unités de ressource doivent être exécutées avant* $t_1$. *Donc* $lst_1$ *peut être ajustée à* $2 - \frac{1}{5}(20 - 15) = 1$.

*De plus,* $let_1$ *peut être ajusté à* $t_1 + 1/r_i^{min} \times (R(t_2 - t_1) - \sum_{j \in A;\ j \neq i} \underline{b}(j, t_1, t_2)) = 2 + (15 - 13) = 4$. *En effet, si l'activité* 1 *finit après* $t_2$, *alors elle doit consommer au moins* $\min(\underline{b}_{RS}(1, 2, 5), \underline{b}_{CS}(1, 2, 5)) = \min(11, 6) = 6$ *unités de ressource dans l'intervalle* $[2, 5[$. *Or, seulement* 2 *unités sont disponibles. L'activité* 1 *ne peut donc pas finir après* $t_2$ *et nous pouvons ajuster* $let_1$ *à* 4.

L'algorithme 2 présente les ajustements pour le CECSP.

Nous avons montré qu'il était possible de calculer, étant donné un intervalle et une activité, sa consommation minimale et les ajustements pouvant être faits sur ses fenêtres de temps en $O(1)$. Étant

**Algorithme 2 :** Ajustement des fenêtres de temps pour le CECSP.

**pour tous les** *intervalles d'intérêt* $[t_1, t_2[$ **faire**
  $W \leftarrow 0$
  $B \leftarrow 0$ **pour tous les** $i \in \mathcal{A}$ **faire**
    $W \leftarrow W + \min(\underline{w}_{LS}(i, t_1, t_2), \underline{w}_{CS}(i, t_1, t_2), \underline{w}_{RS}(i, t_1, t_2))$
    $B \leftarrow B + \max(r_i^{min} \frac{W}{f_i(r_i^{min})}, \frac{1}{a_i}(W - c_i|I|))$
  **si** $B > R(t_2 - t_1)$ **alors**
    L'instance est infaisable.
  **sinon**
    **pour tous les** $i \in A$ **faire**
      $W \leftarrow \min(\underline{w}_{LS}(i, t_1, t_2), \underline{w}_{CS}(i, t_1, t_2), \underline{w}_{RS}(i, t_1, t_2))$
      $slack \leftarrow R(t_2 - t_1) - B + \max(r_i^{min} \frac{W}{f_i(r_i^{min})}, \frac{1}{a_i}(W - c_i|I|))$ **si** $slack < \underline{b}_{LS}(i, t_1, t_2)$ **alors**
        $e_i^{min} \leftarrow \max(e_i^{min}, t_2 + \frac{1}{r_i^{max}}(\underline{b}_{LS}(i, t_1, t_2) - slack))$
      **si** $slack < \min(\underline{b}_{LS}(i, t_1, t_2), \underline{b}_{CS}(i, t_1, t_2))$ **alors**
        $est_i \leftarrow \max(est_i, t_2 - \frac{slack}{r_i^{min}})$
      **si** $slack < \underline{b}_{RS}(i, t_1, t_2)$ **alors**
        $s_i^{max} \leftarrow \min(s_i^{max}, t_1 - \frac{1}{r_i^{max}}(\underline{b}_{RS}(i, t_1, t_2) - slack))$
      **si** $slack < \min(\underline{b}_{RS}(i, t_1, t_2), \underline{b}_{CS}(i, t_1, t_2))$ **alors**
        $let_i \leftarrow \min(let_i, t_1 + \frac{slack}{r_i^{min}})$

donné un intervalle, la fonction de marge ainsi que tous les ajustements peuvent donc être calculés en $O(n)$. Dans la sous-section suivante, nous montrons qu'il suffit d'exécuter l'algorithme de vérification, ainsi que les ajustements sur un nombre polynomial d'intervalles $[t_1, t_2[$.

### 3.2.3 Caractérisation des intervalles d'intérêt

Dans un premier temps, nous prouvons que nous pouvons seulement considérer un nombre polynomial d'intervalles pour détecter une incohérence. En effet, à cause de la nature continue du problème, on aurait pu être amené à considérer un nombre potentiellement infini d'intervalles.

**Théorème 3.2.** *L'algorithme de vérification du raisonnement énergétique a seulement besoin d'être appliqué sur un nombre polynomial d'intervalles* $[t_1, t_2[$.

*Démonstration.* La fonction de marge étant la différence entre une fonction affine, $R(t_2 - t_1)$, et la somme de fonction affine par morceaux, $\underline{b}(i, t_1, t_2)$, c'est aussi une fonction affine par morceaux à deux dimensions. Le minimum de cette fonction est donc atteint en un point extrême d'un des polygones convexes dans lequel elle est affine.

Les segments de définitions, i.e. les segments où l'expression de la fonction est la même, de la fonction de marge sont les mêmes que ceux de la somme des consommations individuelles de chaque activité. Donc, d'un point de vue géométrique, un point extrême de la fonction de marge est l'intersection de deux segments chacun correspondant à un segment de définition de la fonction de consommation d'une activité.

Pour calculer la fonction de marge, il suffit donc d'énumérer tous ces points d'intersections et, pour chacun d'entre eux, d'appliquer le test de satisfiabilité décrit par le théorème 3.1. Comme, pour chaque activité, il existe un nombre constant de segments de définition, le nombre de points d'intersection est de l'ordre de $O(n^2)$. □

De la même manière, nous pouvons prouver que les ajustements peuvent être appliqués sur un nombre polynomial d'intervalles. Pour cela, il suffit de considérer, à la place de la fonction de marge, la fonction $R(t_2-t_1)-\sum_{\substack{j\in\mathcal{A}\\ j\neq i}}\underline{b}(j,t_1,t_2)-\underline{b}_{RS}(i,t_1,t_2)$ pour la règle 3.4 et $R(t_2-t_1)-\sum_{\substack{j\in\mathcal{A}\\ j\neq i}}\underline{b}(j,t_1,t_2)-\min(\underline{b}_{CS}(i,t_1,t_2),\underline{b}_{LS}(i,t_1,t_2))$ pour la règle 3.5. Dans la suite, nous appellerons ces fonctions *fonction d'ajustement de* $lst_i$ *et* $est_i$ respectivement.

**Théorème 3.3.** *L'algorithme de filtrage du raisonnement énergétique a seulement besoin d'être appliqué sur un nombre polynomial d'intervalles* $[t_1,t_2[$.

Dans les paragraphes suivants, nous allons présenter trois méthodes permettant de calculer les intervalles d'intérêt du raisonnement énergétique. Les deux premières méthodes s'appuient sur une analyse des segments de définition des fonctions de consommation individuelle des activités tandis que la seconde est une adaptation du travail de [DP14] pour la contrainte cumulative et se base sur une analyse des variations des fonctions de marge et d'ajustements.

#### Analyse des segments de définition des fonctions de consommation individuelle

Dans ce paragraphe, nous allons donc étudier les segments de définitions des fonctions de consommation individuelle en fonctions des caractéristiques des activités. Plus précisément, nous devons considérer trois cas :

- $r_i^{min}=0$
- $W_i \leq f_i(r_i^{min})(let_i-est_i)$
- $W_i \geq f_i(r_i^{min})(let_i-est_i)$

Ces trois cas devront être considérés séparément dans chacune des méthodes proposées pour calculer les intervalles d'intérêt du raisonnement énergétique (pour l'algorithme de vérification et les ajustements). Cependant, seul le troisième cas sera détaillé dans le manuscrit, les deux autres cas étant traités de manière similaire.

Ce raisonnement peut aussi s'appliquer pour calculer les intervalles d'intérêt pour les ajustements mais ceci ne sera pas détaillé dans ce manuscrit, cette méthode n'étant pas la plus efficace. De même, nous allons seulement présenter cette technique dans le cas des fonctions affines mais elle peut être étendues facilement au cas des fonctions concaves et affines par morceaux. La caractérisation des intervalles d'intérêt dans ces deux cas sera présentée dans le paragraphe suivant et calculée à l'aide de l'analyse de la fonction de marge et des fonctions d'ajustements.

Pour analyser les segments de définition des fonctions de consommation individuelle, nous allons, dans un premier temps, montrer que si nous analysons seulement les segments de définition de $\underline{b}(i,t_1,t_2)$ pour $t_1 \geq est_i$, $t_2 \leq let_i$ et $t_1+t_2 \leq est_i+let_i$, alors le reste des segments peut être déduit par symétrie.

**Lemme 3.1.**

$$\underline{b}(i,t_1,t_2)=\underline{b}(i,est_i,t_2),\ \forall t_1,t_2\in\mathbb{R},\ t_1\leq est_i$$

$$\underline{b}(i,t_1,t_2) = \underline{b}(i,t_1,let_i),\ \forall t_1,t_2 \in \mathbb{R},\ t_2 \geq let_i$$

*Démonstration.* Pour tout intervalle $[t_1,t_2[$ tel que $t_1 \leq est_i$, nous avons :

$$\int_{t_1}^{t_2} b_i(t)dt = \int_{t_1}^{est_i} b_i(t)dt + \int_{est_i}^{t_2} b_i(t)dt$$

Or, $b_i(t) = 0, \forall t \leq est_i$. Nous avons donc que :

$$\int_{t_1}^{est_i} b_i(t)dt + \int_{est_i}^{t_2} b_i(t)dt = \int_{est_i}^{t_2} b_i(t)dt$$

De plus, d'après l'expression de la consommation minimale (3.9), nous avons aussi que :

$$\begin{aligned}
\underline{b}(i,t_1,t_2) &= \min \int_{t_1}^{t_2} b_i(t)dt \\
&= \min \int_{t_1}^{est_i} b_i(t)dt + \min \int_{est_i}^{t_2} b_i(t)dt \\
&= \min \int_{est_i}^{t_2} b_i(t)dt \\
&= \underline{b}(i,est_i,t_2)
\end{aligned}$$

Nous pouvons montrer de la même manière la seconde égalité et ceci achève la démonstration. □

**Lemme 3.2.** *Pour tout* $t_1 \geq 0,\ t_2 \geq t_1,\ \underline{b}(i,t_1,t_2) = \underline{b}(i,let_i + est_i - t_2, let_i + est_i - t_1)$.

*Démonstration.* Pour montrer le lemme, on peut montrer que $\underline{w}(i,t_1,t_2) = \underline{w}(i,let_i + est_i - t_2, let_i + est_i - t_1)$. En effet, pour une même quantité d'énergie, la conversion de $\underline{w}(i,t_1,t_2)$ en $\underline{b}(i,t_1,t_2)$ ne dépend que de la taille de l'intervalle et $let_i + est_i - t_1 - let_i - est_i + t_2 = t_2 - t_1$. Nous montrons donc le lemme pour $\underline{w}(i,t_1,t_2)$.

— $\underline{w}_{LS}(i,t'_1,t'_2) = \max(0, W_i - f_i(r_i^{max})(\max(0, let_i + est_i - t_2 - est_i))) = \underline{w}_{RS}(i,t_1,t_2)$

— $\underline{w}_{RS}(i,t'_1,t'_2) = \max(0, W_i - f_i(f_i(r_i^{max}))(\max(0, let_i - let_i - est_i + t_1))) = \underline{w}_{LS}(i,t_1,t_2)$

$$\underline{w}_{CS}(i,t'_1,t'_2) = \max \left( \begin{array}{c} f_i(r_i^{min})(let_i + est_i - t_1 - est_i - let_i + t_2), \\ W_i - f_i(r_i^{max})(\max(0, let_i - let_i - est_i + t_1 + \\ let_i + est_i - t_2 - est_i) \end{array} \right)$$

—

$$\begin{aligned}
&= \max \left( \begin{array}{l} f_i(r_i^{min})(t_2 - t_1), \\ W_i - f_i(r_i^{max})(\max(0, let_i - t_2 + t_1 - est_i)) \end{array} \right) \\
&= \underline{w}_{CS}(i,t_1,t_2)
\end{aligned}$$

Et donc :

$$\begin{aligned}
\underline{w}(i,t'_1,t'_2) &= \min(\underline{w}_{LS}(i,t'_1,t'_2), \underline{w}_{CS}(i,t'_1,t'_2), \underline{w}_{RS}(i,t'_1,t'_2)) \\
&= \min(\underline{w}_{RS}(i,t_1,t_2), \underline{w}_{CS}(i,t_1,t_2), \underline{w}_{LS}(i,t_1,t_2)) \\
&= \underline{w}(i,t_1,t_2)
\end{aligned}$$

avec $t'_1 = let_i + est_i - t_2$ et $t'_2 = let_i + est_i - t_1,\ \forall t_1 \leq 0$ et $\forall t_2 \geq t_1$. □

Donc, grâce aux lemmes 3.1 et 3.2, nous avons seulement besoin d'établir l'expression de $\underline{b}(i, t_1, t_2)$ à l'intérieur du polygone (triangle) délimité par les inégalités $t_1 \geq est_i,\ t_2 \leq let_i,\ t_1 + t_2 \leq est_i + let_i$ et $t_2 \geq t_1$. Ce triangle est défini par les points :

$$A = (est_i, let_i) \qquad B = (\tfrac{est_i + let_i}{2}, \tfrac{est_i + let_i}{2}) \qquad C = (est_i, est_i)$$

Les différentes expressions de $\underline{w}(i, t_1, t_2)$ en fonction de $t_1, t_2$ et des paramètres de l'activité $i$ sont décrits dans [AL15]. Nous commençons donc par brièvement présenter ces résultats, puis nous expliquerons comment les étendre pour obtenir les différentes expressions de $\underline{b}(i, t_1, t_2)$.

Pour ce faire et pour simplifier les calculs, nous introduisons les deux notations suivantes : $s_i^{max}$ est la borne supérieure $let_i - W_i/f_i(r_i^{max})$ sur la date de début de $i$ et $e_i^{min}$ la borne inférieure $est_i + W_i/f_i(r_i^{max})$ sur la date de fin de $i$.

Afin de décrire les zones où l'expression de $\underline{w}(i, t_1, t_2)$ est identique, il faut analyser les conditions sur $t_1$ et $t_2$ selon lesquelles une expression est choisie plutôt qu'une autre. Pour cela, les auteurs de [AL15] se proposent d'étudier les segments correspondant au cas où deux expressions ont la même valeur, e.g. $f_i(r_i^{min})(t_2 - t_1) = W_i - f_i(r_i^{max})(t_1 - est_i)$. Ces segments permettent de délimiter des zones dans lesquelles on doit décider l'expression de $\underline{w}(i, t_1, t_2)$, i.e. quelle expression est minimale à l'intérieur de cette zone.

Par exemple, dans le cas où $W_i \geq f_i(r_i^{min})(let_i - est_i)$ l'unique point pour lequel l'énergie requise est maximale est le point $A$. Donc, $\underline{w}(i, est_i, let_i) = W_i$ et, dans la zone délimitée par les lignes $t_1 \leq est_i$ et $t_2 \geq let_i$ (en rouge sur la figure 3.11), l'expression de $\underline{w}(i, t_1, t_2)$ est $W_i$.

Notons aussi que le cas $W_i \geq f_i(r_i^{min})(let_i - est_i)$ correspond au cas où l'activité ne peut être ordonnancée à $r_i^{min}$ durant toute sa durée d'exécution. C'est-à-dire que l'énergie requise par l'activité $i$ dans l'intervalle $[t_1, t_2[$ peut dépasser $f_i(r_i^{min})(t_2 - t_1)$ et le cas où $\underline{w}(i, t_1, t_2) = W_i - f_i(r_i^{max})(let_i - t_2 + t_1 - est_i)$ doit être considéré. De plus, le cas $W_i \geq f_i(r_i^{min})(let_i - est_i)$ est séparé en deux sous cas : $e_i^{min} \leq s_i^{max}$ et $e_i^{min} \geq s_i^{max}$. Le premier cas correspond au cas où le point d'intersection des lignes $t_1 \leq e_i^{min}$ et $t_2 \geq s_i^{max}$ vérifie $t_2 \geq t_1$ et donc le point est à l'intérieur du triangle $(A, B, C)$. Le second cas correspond au cas où ce point se trouve à l'extérieur du triangle. Dans ces deux cas, les segments de définition à considérer ne sont pas les mêmes.

Plaçons nous dans le premier cas, i.e. le cas où $e_i^{min} \leq s_i^{max}$. Dans ce cas, le segment $\overline{DD'}$ sépare les cas où $\underline{w}(i, t_1, t_2) = f_i(r_i^{min})(t_2 - t_1)$ et $\underline{w}(i, t_1, t_2) = W_i - f_i(r_i^{max})(let_i - t_2 + t_1 - est_i)$, le segment $\overline{DG}$ délimite quant à lui les zones où $\underline{w}(i, t_1, t_2) = f_i(r_i^{min})(t_2 - t_1)$ et $\underline{w}(i, t_1, t_2) = W_i - f_i(r_i^{max})(let_i - t_2)$. Les autres zones peuvent être définies de manière similaire. Pour une analyse complète des différents expressions de $\underline{w}(i, t_1, t_2)$, nous renvoyons le lecteur à [AL15].

La figure 3.11 présente ces différentes expressions avec les expressions de $\underline{w}(i, t_1, t_2)$ suivantes :

— la zone rouge correspond à $\underline{w}(i, t_1, t_2) = W_i$ ;
— la zone verte à $\underline{w}(i, t_1, t_2) = W_i - (let_i - t_2) f_i(r_i^{max})$ ;
— la zone bleue à $\underline{w}(i, t_1, t_2) = W_i - (t_1 - est_i) f_i(r_i^{max})$ ;
— la zone jaune à $\underline{w}(i, t_1, t_2) = W_i - (let_i - t_2 + t_1 - est_i) f_i(r_i^{max}))$ ;
— et la blanche à $\underline{w}(i, t_1, t_2) = (t_2 - t_1) f_i(r_i^{min})$.

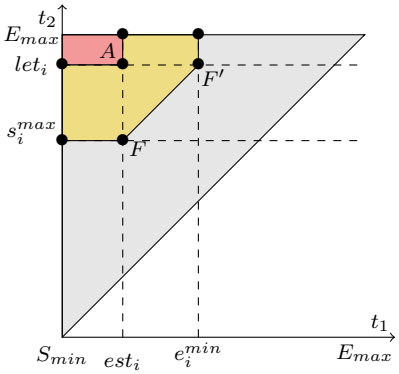
Les autres zones correspondent à $\underline{w}(i, t_1, t_2) = 0$.

De plus, les coordonnées de chaque point sont :

— $A = (est_i, let_i)$, $B = (e_i^{min}, s_i^{max})$, $C = (s_i^{max}, s_i^{max})$ and $C' = (e_i^{min}, e_i^{min})$

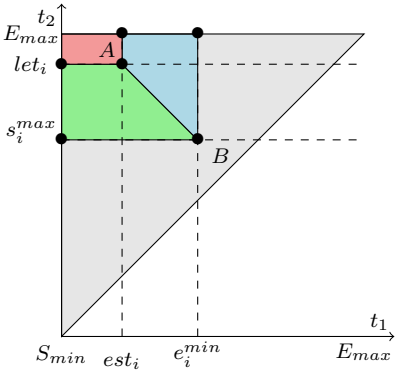
— $D = (est_i, \frac{let_i f_i(r_i^{max}) - est_i f_i(r_i^{min}) - W_i}{f_i(r_i^{max}) - f_i(r_i^{min})})$, $D' = (\frac{est_i f_i(r_i^{max}) - let_i f_i(r_i^{min}) + W_i}{f_i(r_i^{max}) - f_i(r_i^{min})}, let_i)$

— $G = (\frac{est_i(f_i(r_i^{max}) - f_i(r_i^{min})) - let_i f_i(r_i^{min}) + W_i}{f_i(r_i^{max}) - 2f_i(r_i^{min})}, \frac{let_i(f_i(r_i^{max}) - f_i(r_i^{min})) - est_i f_i(r_i^{min}) - W_i}{f_i(r_i^{max}) - 2f_i(r_i^{min})})$

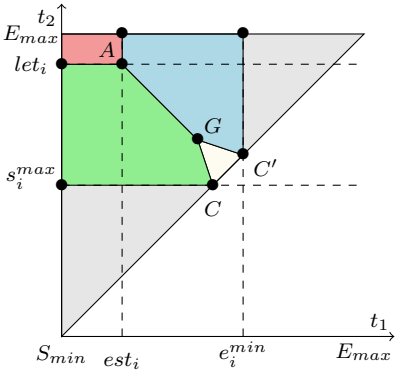



(a) Cas $r_i^{min} = 0$

Cas $W_i \leq f_i(r_i^{min})(let_i - est_i)$

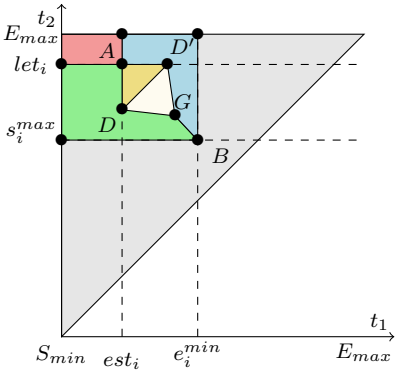



(b) $e_i^{min} \leq s_i^{max}$

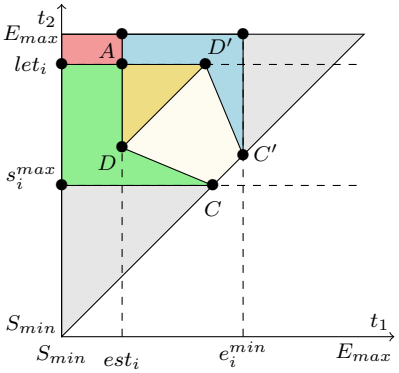



(c) $e_i^{min} \geq s_i^{max}$

Cas $W_i \geq f_i(r_i^{min})(let_i - est_i)$




(d) $e_i^{min} \leq s_i^{max}$




(e) $e_i^{min} \geq s_i^{max}$

FIGURE 3.11 – Les différentes expressions de $\underline{w}(i, t_1, t_2)$ en fonction des paramètres du problème CECSP.

Nous avons donc défini les différentes zones correspondant chacune à une expression de $\underline{w}(i,t_1,t_2)$. Dans le cas où $f_i$ est la fonction identité, cela suffit pour définir les segments à considérer, i.e. les segments pour lesquels on va tester l'intersection. La méthode générale pour calculer les intervalles d'intérêt en fonction de ces segments sera détaillée un peu plus loin dans le paragraphe.

Les segments à considérer sont les segments délimitant deux zones avec une expression de $\underline{w}(i,t_1,t_2)$ différentes, i.e. reliant deux points représentés par une boule noire sur la figure 3.11. Par exemple, dans le cas où $W_i \geq f_i(r_i^{min})(let_i - est_i)$ les segments à considérer sont donc :

— dans le cas où $e_i^{min} \leq s_i^{max}$ : $(est_i, E_{max})A$, $(S_{min}, let_i)A$, $(S_{min}, s_i^{max})B$, $B(e_i^{min}, E_{max})$, $AD$, $AD'$, $DD'$, $DG$, $D'G$ et $GB$.

— dans le cas où $e_i^{min} \geq s_i^{max}$ : $(est_i, E_{max})A$, $(S_{min}, let_i)A$, $(S_{min}, s_i^{max})C$, $C'(e_i^{min}, E_{max})$, $AD$, $AD'$, $DD'$, $DC$ et $D'C$.

avec $D_{t_1}$ (resp. $D_{t_2}$) la projection sur l'axe $x$ (resp. sur l'axe $y$) du point $D$.

Nous allons maintenant calculer, à partir de la figure 3.11, les zones correspondant à des expressions différentes de $\underline{b}(i,t_1,t_2)$.

Pour ce faire, nous considérons, à l'intérieur de chaque zone définie pour $\underline{w}(i,t_1,t_2)$, l'inégalité suivante :

$$\frac{\underline{w}(i,t_1,t_2)}{f_i(r_i^{min})} \leq |I|$$

avec $I = [t_1,t_2[\cap[est_i, let_i]$. Si cette condition est vérifiée, alors l'activité peut être ordonnancée à $r_i^{min}$ et l'expression de $\underline{b}(i,t_1,t_2)$ sera alors $\frac{\underline{w}(i,t_1,t_2)}{f_i(r_i^{min})} \times r_i^{min}$. Dans le cas inverse, nous avons $\underline{b}(i,t_1,t_2) = \frac{1}{a_i}(\underline{w}(i,t_1,t_2) - c_i|I|)$.

Dans le cas où $r_i^{min} = 0$, l'inégalité n'a pas besoin d'être considérée et nous avons directement $\underline{b}(i,t_1,t_2) = \frac{1}{a_i}(\underline{w}(i,t_1,t_2) - c_i|I|)$ dans chaque zone.

Dans le cas où $W_i \leq f_i(r_i^{min})(let_i - est_i)$, $|I|$ peut valoir $t_2 - t_1$, $t_2 - est_i$, $let_i - t_1$ ou $let_i - est_i$.

Dans le premier cas, l'inégalité donne $\underline{w}(i,t_1,t_2) \leq f_i(r_i^{min})(t_2 - t_1)$. Or, ce cas correspond au cas où $[t_1,t_2[\subset [est_i, let_i]$ et donc, $\underline{w}(i,t_1,t_2) = \min(W_i - f_i(r_i^{max})(let_i - t_2), W_i - f_i(r_i^{max})(t_1 - est_i), f_i(r_i^{min})(t_2 - t_1))$. Donc, dans ce cas, l'inégalité est toujours vérifiée et nous avons $\underline{b}(i,t_1,t_2) = r_i^{min} \times \underline{w}(i,t_1,t_2)/f_i(r_i^{min})$.

Dans le second cas, i.e. $|I| = t_2 - est_i$, l'inégalité donne $\underline{w}(i,t_1,t_2) \leq f_i(r_i^{min})(t_2 - est_i)$. Or, ce cas correspond au cas où $\underline{w}(i,t_1,t_2) = W_i - f_i(r_i^{max})(let_i - t_2)$ , i.e. la zone verte. En remplaçant $\underline{w}(i,t_1,t_2)$ par son expression, nous obtenons l'inégalité suivante :

$$t_2 \leq \frac{f_i(r_i^{max})let_i - f_i(r_i^{min})est_i - W_i}{f_i(r_i^{max}) - f_i(r_i^{min})}$$

Or, $let_i \leq \frac{f_i(r_i^{max})let_i - f_i(r_i^{min})est_i - W_i}{f_i(r_i^{max}) - f_i(r_i^{min})}$. Donc, l'inégalité est vérifiée si $t_2 \leq d_i$ et ceci est vrai dans ce cas.

Les cas où $|I| = let_i - est_i$ et $|I| = let_i - t_1$ sont traités de la même façon. Donc, dans le cas où $W_i \leq f_i(r_i^{min})(let_i - est_i)$, $\underline{b}(i,t_1,t_2) = r_i^{min} \times \underline{w}(i,t_1,t_2)/f_i(r_i^{min})$ et les zones où $\underline{b}(i,t_1,t_2)$ a la même expression sont les mêmes que pour $\underline{w}(i,t_1,t_2)$ dans ce cas.

Dans le cas où $W_i \geq f_i(r_i^{max})(let_i - est_i)$, si $|I|$ vaut $t_2 - t_1$, l'inégalité donne $\underline{w}(i,t_1,t_2) \leq f_i(r_i^{min})(t_2-t_1)$. Donc, si $\underline{w}_{CS}(i,t_1,t_2) = f_i(r_i^{min})(t_2-t_1)$ alors $\underline{w}(i,t_1,t_2) = \min(W_i - f_i(r_i^{max})(let_i - t_2), W_i - f_i(r_i^{max})(t_1 - est_i), f_i(r_i^{min})(t_2-t_1))$ et l'inégalité est vérifiée. A l'inverse, si $\underline{w}_{CS}(i,t_1,t_2) = W_i - f_i(r_i^{max})(let_i - t_2 + t_1 - est_i)$ et comme $[t_1,t_2[ \subset [est_i, let_i]$, on a forcément $\underline{w}(i,t_1,t_2) = W_i - f_i(r_i^{max})(let_i - t_2 + t_1 - est_i)$. Donc l'inégalité n'est jamais vérifiée et $\underline{b}(i,t_1,t_2) = \frac{1}{a_i}(\underline{w}(i,t_1,t_2) - c_i|I|)$.

Le cas où $|I|$ vaut $t_2 - est_i$ correspond au cas où $\underline{w}(i,t_1,t_2) = W_i - f_i(r_i^{max})(let_i - t_2)$. Dans ce cas, l'inégalité donne :

$$\frac{let_i f_i(r_i^{max}) - est_i f_i(r_i^{min}) - W_i}{f_i(r_i^{max}) - f_i(r_i^{min})} \leq t_2$$

La partie gauche de cette inégalité correspond exactement à l'ordonnée du point $D$ et donc la zone verte est divisée en deux parties :

— la partie claire où $\underline{b}(i,t_1,t_2) = r_i^{min} \times \underline{w}(i,t_1,t_2)/f_i(r_i^{min})$

— et la partie plus foncée où $\underline{b}(i,t_1,t_2) = \frac{1}{a_i}(\underline{w}(i,t_1,t_2) - c_i|I|)$.

Les cas où $|I| = let_i - est_i$ et $|I| = let_i - t_1$ sont traités de la même façon.

La figure 3.12 présente ces résultats avec les expressions de $\underline{b}(i,t_1,t_2)$ suivantes :

— la zone rouge claire correspond à $\underline{b}(i,t_1,t_2) = r_i^{min} \times W_i / f_i(r_i^{min})$ ;

— la zone rouge foncée correspond à $\underline{b}(i,t_1,t_2) = \frac{1}{a_i}(W_i - c_i(let_i - est_i))$ ;

— la zone verte claire à $\underline{b}(i,t_1,t_2) = r_i^{min} \times \{W_i - (let_i - t_2)f_i(r_i^{max})\} / f_i(r_i^{min})$ ;

— la zone verte foncée à $\underline{b}(i,t_1,t_2) = \frac{1}{a_i}(W_i - (let_i - t_2)f_i(r_i^{max}) - c_i|I|)$ où $I$ vaut $[t_1,t_2[$ ou $[est_i,t_2[$ ;

— la zone bleue claire à $\underline{b}(i,t_1,t_2) = r_i^{min} \times \{W_i - (t_1 - est_i)f_i(r_i^{max})\} / f_i(r_i^{min})$ ;

— la zone bleue foncée à $\underline{b}(i,t_1,t_2) = \frac{1}{a_i}(W_i - (t_1 - est_i)f_i(r_i^{max}) - c_i|I|)$ où $I$ vaut $[t_1,t_2[$ ou $[t_1,let_i[$ ;

— la zone jaune à $\underline{b}(i,t_1,t_2) = \frac{1}{a_i}(W_i - (let_i - t_2 + t_1 - est_i)f_i(r_i^{max}) - c_i(t_2 - t_1))$ ;

— et la blanche à $\underline{b}(i,t_1,t_2) = (t_2 - t_1)r_i^{min}$.

Les autres zones correspondent à $\underline{b}(i,t_1,t_2) = 0$.

Les segments à considérer sont aussi décrits par la figure 3.12 et sont, par exemple, dans le cas où $W_i \geq f_i(r_i^{min})(let_i - est_i)$ :

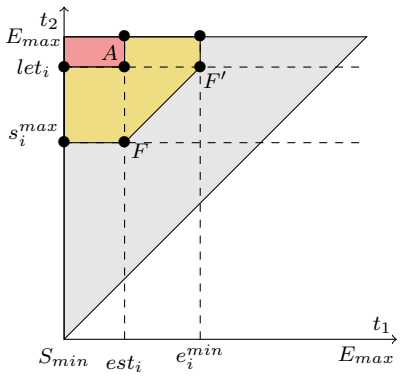
— si $e_i^{min} \leq s_i^{max}$ : $(est_i, E_{max})A$, $(S_{min}, let_i)A$, $(S_{min}, s_i^{max})B$, $B(e_i^{min}, E_{max})$, $AD$, $AD'$, $D(0,D_{t_2})$, $D'(D'_{t_1}, E_{max})$, $DD'$, $DG$, $D'G$ et $GB$.

— si $e_i^{min} \geq s_i^{max}$ : $(est_i, E_{max})A$, $(S_{min}, let_i)A$, $(S_{min}, s_i^{max})C$, $C'(e_i^{min}, E_{max})$, $AD$, $AD'$, $D(0,D_{t_2})$, $D'(D'_{t_1}, E_{max})$, $DD'$, $DC$ et $D'C$.

avec $E_{max} = \max_{i \in \mathcal{A}}\{let_i\}$ et $S_{min} = \min_{i \in \mathcal{A}}\{est_i\}$.

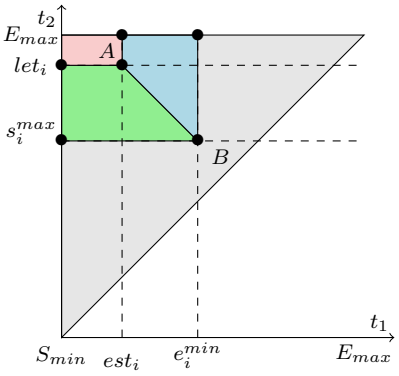
Pour calculer les intervalles d'intérêt pour l'algorithme de vérification du raisonnement énergétique, il faut, pour chaque paire de segments de définition, calculer leur point d'intersection, s'il existe. Les coordonnées de ce point d'intersection correspondent à un intervalle d'intérêt.

Pour calculer ces points d'intersection, deux méthodes existent. La première est la plus naïve et consiste à considérer toutes les paires de segments et calculer leur intersection. La seconde méthode utilise l'algorithme de balayage de Bentley-Ottmann [BO79]. L'idée principale sur laquelle repose l'algorithme de balayage est que deux segments ne peuvent s'intersecter si leurs projections sur l'axe des ordonnées et sur l'axe des abscisses ne se chevauchent pas. Une ligne horizontale fictive est utilisée

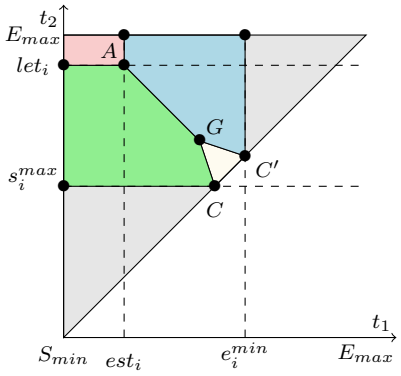



(a) Cas $r_i^{min} = 0$

Cas $W_i \leq f_i(r_i^{min})(let_i - est_i)$

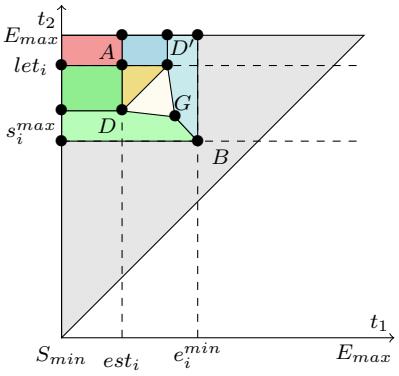



(b) $e_i^{min} \leq s_i^{max}$

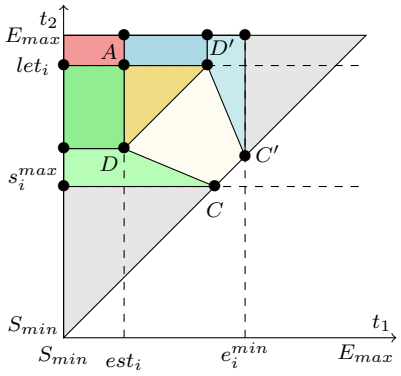



(c) $e_i^{min} \geq s_i^{max}$

Cas $W_i \geq f_i(r_i^{min})(let_i - est_i)$




(d) $e_i^{min} \leq s_i^{max}$




(e) $e_i^{min} \geq s_i^{max}$

FIGURE 3.12 – Les différentes expressions de $\underline{b}(i, t_1, t_2)$ en fonction des paramètres du problème CECSP.

pour balayer l'axe des abscisses et à certains "événements", on teste l'intersection entre deux segments s'ils intersectent tous les deux cette ligne fictive et s'ils sont consécutifs dans l'ordre vertical. Avec cet algorithme, le nombre de tests d'intersection peut grandement décroître par rapport à l'algorithme naïf.

Dans le cas de l'algorithme naïf, la complexité totale de l'algorithme de vérification est de $O(n^3)$ tandis que, pour l'algorithme de balayage la complexité est de $O((n^2+nk)\log n)$, avec $k$ le nombre de points d'intersection. En théorie, la complexité du second algorithme peut être plus élevée ($k$ peut être de l'ordre de $O(n^2)$), mais en pratique, cette complexité peut devenir très faible si le nombre de points d'intersection est petit. Une comparaison de ces deux algorithmes sera effectuée dans le chapitre 6.

**Analyses des variations de la fonction de marge et des fonctions d'ajustements**

Nous allons décrire ici une troisième méthode permettant le calcul des intervalles d'intérêt du raisonnement énergétique. Cette méthode est l'adaptation des travaux de [DP14] dans le cadre du CuSP. Dans un premier temps, nous décrivons la méthode permettant de calculer ces intervalles pour l'algorithme de vérification, puis nous présenterons ce même résultat dans le cadre des ajustements de fenêtres de temps.

La méthode décrite dans ce paragraphe repose sur le fait que, pour chercher un intervalle $[t_1,t_2[$ vérifiant $SL(t_1,t_2)<0$, il suffit de s'intéresser au point $(t_1,t_2)$ pour lesquels la fonction de marge est minimale. Comme, en pratique, il est difficile de trouver le minimum global d'une telle fonction, nous nous intéressons à ces minima locaux. Grâce au test de la dérivée seconde (voir sous-section 2.2.3), nous pouvons dériver des conditions selon lesquels $SL$ est localement minimale.

**Lemme 3.3** ([DP14])**.** *La fonction de marge $SL(t_1,t_2)$ est localement minimale seulement s'il existe deux activités $i$ et $j$ telles que les conditions ci-dessous sont satisfaites :*

$$\frac{\delta^- \underline{b}(i,t_1,t_2)}{\delta t_1} > \frac{\delta^+ \underline{b}(i,t_1,t_2)}{\delta t_1} \tag{3.25}$$

$$\frac{\delta^- \underline{b}(j,t_1,t_2)}{\delta t_2} > \frac{\delta^+ \underline{b}(j,t_1,t_2)}{\delta t_2} \tag{3.26}$$

Le preuve est identique à celle du lemme 2.1. Le lemme 3.3 peut être utilisé pour déterminer les conditions nécessaires permettant de déterminer l'ensemble des intervalles d'intérêt. Pour cela, une étude des fonctions $t_1 \to \underline{b}(i,t_1,t_2)$ et $t_2 \to \underline{b}(i,t_1,t_2)$ est nécessaire.

**Théorème 3.4.** *Pour chaque activité $i$ et pour tout début d'intervalle $t_1$ il existe au plus deux intervalles $[t_1,t_2[$ tels que $\frac{\delta^- \underline{b}(i,t_1,t_2)}{\delta t_2} > \frac{\delta^+ \underline{b}(i,t_1,t_2)}{\delta t_2}$. De même, pour chaque activité $i$ et pour toute fin d'intervalle $t_2$, il existe au plus deux intervalles $[t_1,t_2[$ tels que $\frac{\delta^- \underline{b}(i,t_1,t_2)}{\delta t_1} > \frac{\delta^+ \underline{b}(i,t_1,t_2)}{\delta t_1}$. Ces intervalles sont décrits par le tableau 3.1 avec :*

$$\Delta'(t_2) = \frac{t_2(f_i(r_i^{min})-f_i(r_i^{max}))+let_i f_i(r_i^{max})-W_i}{f_i(r_i^{min})} \quad ; \quad \Gamma'(t_2) = \frac{W_i - t_2 f_i(r_i^{min})+est_i f_i(r_i^{max})}{f_i(r_i^{max})-f_i(r_i^{min})}$$

*les points de cassures de la fonction $t_1 \to \underline{b}(i,t_1,t_2)$ et*

$$\Gamma(t_1) = \frac{W_i - t_1(f_i(r_i^{min})-f_i(r_i^{max}))+est_i f_i(r_i^{max})}{f_i(r_i^{min})} \quad ; \quad \Delta(t_1) = \frac{W_i - f_i(r_i^{min})let_i + t_1 f_i(r_i^{max})}{f_i(r_i^{max})-f_i(r_i^{min})}$$

*de la fonction $t_2 \to \underline{b}(i,t_1,t_2)$.*

Notons que le cas $r_i^{min} = r_i^{max}$, i.e. le cas cumulatif, est inclus dans le cas $W_i \leq f_i(r_i^{min})(let_i - est_i)$ et que dans ce cas :

| | | | |
|---|---|---|---|
| $r_i^{min} = 0$ | $t_1 \leq e_i^{min}$ | $t_1 \leq est_i$ | $[t_1, let_i[$ |
| $W_i \leq f_i(r_i^{min})(let_i - est_i)$ | $est_i \leq t_1 \leq e_i^{min}$ | $e_i^{min} \leq s_i^{max}$ xor $t_1 \leq G_{t_1}$ | $[t_1, est_i + let_i - t_1[$ |
| | | $t_1 \geq s_i^{max}$ | $[t_1, \Gamma(t_1)[$ |
| | | $e_i^{min} > s_i^{max}$ et $s_i^{max} > t_1 > G_{t_1}$ | $[t_1, \Gamma(t_1)[$ $[t_1, \Delta(t_1)[$ |
| | $t_1 \leq est_i$ | | $[t_1, let_i[$ |
| $W_i \geq f_i(r_i^{min})(let_i - est_i)$ | $est_i \leq t_1 \leq e_i^{min}$ | $t_1 > D'_{t_1}$ et $t_1 < s_i^{max}$ et $t_1 < G_{t_1}$ | $[t_1, \Gamma(t_1)[$ $[t_1, \Delta(t_1)[$ |
| | | $t_1 \leq D'_{t_1}$ | $[t_1, let_i[$ $[t_1, \Delta(t_1)[$ |
| | | $t_1 \geq s_i^{max}$ et $e_i^{min} \geq s_i^{max}$ | $[t_1, \Gamma(t_1)[$ |
| | | $t_1 \geq G_{t_1}$ et $e_i^{min} < s_i^{max}$ | $[t_1, est_i + let_i - t_1[$ |

(a) $t_2$ vérifiant (3.26)

| | | | |
|---|---|---|---|
| $r_i^{min} = 0$ | $t_2 \geq s_i^{max}$ | $t_2 \geq let_i$ | $[est_i, t_2[$ |
| $W_i \leq f_i(r_i^{min})(let_i - est_i)$ | $let_i \geq t_2 \geq s_i^{max}$ | $e_i^{min} \leq s_i^{max}$ xor $t_2 \geq G_{t_2}$ | $[est_i + let_i - t_2, t_2[$ |
| | | $t_2 \leq e_i^{min}$ | $[\Delta'(t_2), t_2[$ |
| | | $e_i^{min} > s_i^{max}$ et $e_i^{min} < t_2 < G_{t_2}$ | $[\Delta'(t_2), t_2[$ $[\Gamma'(t_2), t_2[$ |
| | $t_2 \geq let_i$ | | $[est_i, t_2[$ |
| $W_i \geq f_i(r_i^{min})(let_i - est_i)$ | $let_i \geq t_2 \geq s_i^{max}$ | $t_2 < D_{t_2}$ et $t_2 > e_i^{min}$ et $t_2 > G_{t_2}$ | $[\Delta'(t_2), t_2[$ $[\Gamma'(t_2), t_2[$ |
| | | $t_2 \geq D_{t_2}$ | $[est_i, t_2[$ $[\Gamma'(t_2), t_2[$ |
| | | $e_i^{min} < s_i^{max}$ et $t_2 \leq e_i^{min}$ | $[\Delta'(t_2), t_2[$ |
| | | $e_i^{min} \geq s_i^{max}$ et $t_2 \leq G_{t_2}$ | $[est_i + let_i - t_2, t_2[$ |

(b) $t_1$ vérifiant (3.25)

TABLE 3.1 – Intervalles d'intérêt pour l'algorithme de vérification du raisonnement énergétique pour le CECSP.

$$\Delta'(t_2) = s_i^{max} \quad \text{et} \quad \Gamma(t_1) = e_i^{min}$$

*Démonstration.* Nous présentons seulement comment nous obtenons les $t_2$ pertinents pour la huitième colonne du tableau 3.1a, tous les autres cas étant obtenus de manière similaire. La signification de cette colonne est la suivante : si $W_i \geq f_i(r_i^{max})(let_i - est_i)$, $est_i \leq t_1 \leq eet_i$ et $t_1 \leq D_{t_1}$, alors les intervalles à considérer sont $[t_1, let_i[$ et $[t_1, \Gamma(t_1)[$.

Pour prouver le lemme, nous analysons les variations de la fonction $t_2 \to \underline{b}(i, t_1, t_2)$. Pour analyser ces variations, nous utilisons les expressions de $\underline{b}(i, t_1, t_2)$ en fonction de $t_1$ et $t_2$ décrites dans le paragraphe précédent. Pour adapter ces zones dans le cas des fonctions concaves et affines par morceaux, il faut déterminer, dans les zones où l'expression de $\underline{b}(i, t_1, t_2)$ est $\frac{1}{a_i}(\underline{w}(i, t_1, t_2) - c_i|I|)$, la valeur de $\max_{p \in \mathcal{P}_i}(\frac{1}{a_{ip}}(\underline{w}(i, t_1, t_2) - c_{ip}|I|))$. Pour ce faire, notons que le segment $\overline{DD'}$ dans la figure 3.12, sépare la zone où $|I| = t_2 - t_1 \geq \underline{w}(i, t_1, t_2)/f_i(r_i^{min})$ de la zone où cette inégalité n'est pas vérifiée. De la même manière, nous pouvons définir une série de segments parallèles , $\overline{D_p D'_p}$, séparant les zones où $|I| = t_2 - t_1 \geq \underline{w}(i, t_1, t_2)/f_i(x_p^i)$ et celle où cette inégalité n'est pas vérifiée, i.e. séparant les zones où $\underline{b}(i, t_1, t_2) = \frac{1}{a_{ip}}(\underline{w}(i, t_1, t_2) - c_{ip}(t_2 - t_1))$. La figure 3.13 représente ces variations.

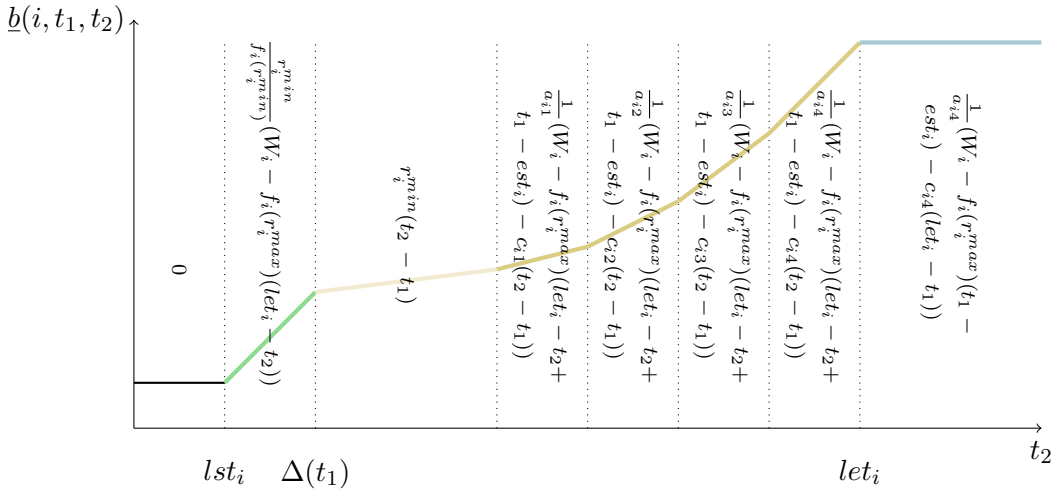



FIGURE 3.13 – Intervalles d'intérêt du raisonnement énergétique du CECSP dans le cas où $est_i < t_1 \leq D'_{t_1}$.

Les intervalles $[t_1, t_2[$ pour lesquels $t_2 \to \underline{b}(i, t_1, t_2)$ vérifie que sa dérivée à gauche est inférieure à sa dérivée à droite sont $[t_1, let_i[$ et $[t_1, \Delta(t_1)[$. En effet, comme $f_i$ est une fonction concave et affine par morceaux, nous avons : $a_{ip} > a_{ip+1}$ et $c_{ip} < c_{ip+1}$, et donc $\frac{\delta^- expr_{ip}}{\delta t_2} < \frac{\delta^+ expr_{ip+1}}{\delta t_2}$.

On peut remarquer que dans la figure 3.13, l'indice de $expr_{ip}$ va seulement jusqu'à 4. Ceci est dû à une simplification. Dans le cas général, l'indice de $expr_{ip}$ s'arrête au rang $\ell$ si $W_i \leq f_i(x_{\ell+1}^i)(let_i - est_i)$. □

En appliquant un raisonnement symétrique pour toute fin d'intervalle $t_2$ fixée, nous obtenons une liste d'intervalles d'intérêt, décrite par le lemme 3.4.

**Lemme 3.4.** *Soient $i$ et $j$ deux activités telles que : $W_l \geq f_l(r_l^{min})(let_l - est_l),\ l = i, j$. Alors, la fonction de marge est localement minimum seulement si $(t_1, t_2)$ correspond à l'un des intervalles suivants :*

$$\begin{cases} [est_j, let_i] & si \Big(est_j \leq est_i \vee (est_j \leq e_i^{min} \wedge est_j \leq D'_{t_1})\Big) \wedge \\ & \Big(let_i \geq let_j \vee (let_i \geq s_j^{max} \wedge let_i \geq D_{t_2})\Big) \\ [\Delta'(let_i), let_i] & si \Big(\Delta'(let_i) \leq est_i \vee (\Delta'(let_i) \leq e_i^{min} \wedge \Delta'(let_i) \leq D'_{t_1})\Big) \wedge \\ & let_j \geq let_i \geq s_j^{max} \wedge let_i \leq D_{t_2} \wedge let_i \geq G_{t_2} \\ [\Gamma'(let_i), let_i] & si \Big(\Gamma'(let_i) \leq est_i \vee (\Gamma'(let_i) \leq e_i^{min} \wedge \Gamma'(let_i) \leq D'_{t_1})\Big) \\ & \wedge let_j \geq let_i \geq s_j^{max} \wedge \Big(let_i \geq e_j^{min} \vee let_i \geq E_{t_2}\Big) \\ [let_j + est_j - let_i, let_i] & si \Big(let_j + est_j - let_i \leq est_i \vee (let_j + est_j - let_i \leq e_i^{min} \wedge \\ & let_j + est_j - let_i \leq D'_{t_1})\Big) \wedge let_j \geq let_i \geq s_j^{max} \wedge let_i \leq G_{t_2} \\ [est_j, \Gamma(est_j)] & si\ est_i \leq est_j \leq e_i^{min} \wedge est_j \geq D'_{t_1} \wedge est_j \leq G_{t_1} \wedge \\ & \Big(\Gamma(est_j) \geq let_j \vee (\Gamma(est_j) \geq s_j^{max} \wedge \Gamma(est_j) \geq D_{t_2})\Big) \\ [est_j, \Delta(est_j)] & si \Big(\Delta(est_j) \geq let_j \vee (\Delta(est_j) \geq s_j^{max} \wedge \Delta(est_j) \geq D_{t_2})\Big) \wedge \\ & est_i \leq est_j \leq e_i^{min} \wedge \Big(s_i^{max} \geq est_j \vee est_j \leq E_{t_1}\Big) \\ [est_j, let_i + est_i - est_j] & si \Big(let_i + est_i - est_j \geq let_j \vee (let_i + est_i - est_j \geq s_j^{max} \\ & \wedge let_i + est_i - est_j \geq D_{t_2})\Big) \wedge est_i \leq est_j \leq e_i^{min} \wedge est_j \geq G_{t_1} \end{cases}$$

Le lemme 3.4 présente les résultats pour le cas où $i$ et $j$ vérifient $W_l \geq f_l(r_l^{min})(let_l - est_l),\ l = i, j$. Les autres cas à considérer sont :

- $r_i^{min} = 0$ et $r_j^{min} = 0$
- $r_i^{min} = 0$ et $W_j \leq f_j(r_j^{min})(let_j - est_j)$
- $r_i^{min} = 0$ et $W_j \geq f_j(r_j^{min})(let_j - est_j)$
- $W_l \leq f_l(r_l^{min})(let_l - est_l),\ l = i, j$
- $W_i \geq f_i(r_i^{min})(let_i - est_i)$ et $W_j \geq f_j(r_j^{min})(let_j - est_j)$

Ces cas peuvent être déduits de manière similaire. De plus, l'algorithme décrit ci-dessous pour appliquer le test de satisfiabilité du raisonnement énergétique n'utilise pas directement cette liste.

Pour décrire cet algorithme, nous introduisons les notations suivantes :

- $O_1 = \{est_i,\ \forall i \in \mathcal{A}\} \cup \{lst_i,\ \forall i \in \mathcal{A} \,|\, r_i^{min} = r_i^{max}\}$
- $O_2 = \{let_i,\ \forall i \in \mathcal{A}\} \cup \{eet_i,\ \forall i \in \mathcal{A} \,|\, r_i^{min} = r_i^{max}\}$

L'algorithme de détection d'incohérence est alors le suivant :

L'algorithme 3 est de complexité $O(n^3)$. En effet, pour chaque couple d'activités $(i, j)$ au plus deux intervalles sont considérés. La complexité est donc la même que dans la première méthode présentée. Cependant, la constante cachée est beaucoup plus petite dans ce cas-là. Une comparaison

**Algorithme 3 :** Algorithme de détection d'incohérence énergétique du CECSP.

**pour tous les** $t_1 \in O_1$ **faire**
- **pour tous les** $i \in \mathcal{A}$ **faire**
  - Calculer $O_{t_2}$ l'ensemble des $t_2$ vérifiant la condition (3.26) à l'aide du tableau 3.1a **pour tous les** $t_2 \in O_{t_2}$ **faire**
    - **si** $R(t_2 - t_1) < \sum_{j \in \mathcal{A}} \underline{b}(j, t_1, t_2)$ **alors**
      - Le problème est infaisable

**pour tous les** $t_2 \in O_2$ **faire**
- **pour tous les** $i \in \mathcal{A}$ **faire**
  - Calculer $O_{t_1}$ l'ensemble des $t_1$ vérifiant la condition (3.25) à l'aide du tableau 3.1b **pour tous les** $t_1 \in O_{t_1}$ **faire**
    - **si** $R(t_2 - t_1) < \sum_{j \in \mathcal{A}} \underline{b}(j, t_1, t_2)$ **alors**
      - Le problème est infaisable

des performances de chaque méthode sera présentée dans le chapitre 6.

Nous allons maintenant montrer comment ce raisonnement peut aussi s'appliquer dans le cas du calcul des intervalles d'intérêt pour les ajustements. Pour ce faire, nous utilisons le lemme 2.4. Comme nous avons déjà calculé les points $t_1$ vérifiant $\frac{\delta^- \underline{b}(i,t_1,t_2)}{\delta t_1} > \frac{\delta^+ \underline{b}(i,t_1,t_2)}{\delta t_1}$ et les points $t_2$ vérifiant $\frac{\delta^- \underline{b}(i,t_1,t_2)}{\delta t_2} > \frac{\delta^+ \underline{b}(i,t_1,t_2)}{\delta t_2}$, il suffit, pour caractériser les intervalles d'intérêt pour les ajustements, de calculer les points $t_1$ vérifiant :

$$\frac{\delta^- \underline{b}_{RS}(i, t_1, t_2)}{\delta t_1} > \frac{\delta^+ \underline{b}_{RS}(i, t_1, t_2)}{\delta t_1} \tag{3.27}$$

et les points $t_2$ vérifiant

$$\frac{\delta^- \underline{b}_{RS}(i, t_1, t_2)}{\delta t_2} > \frac{\delta^+ \underline{b}_{RS}(i, t_1, t_2)}{\delta t_2} \tag{3.28}$$

Pour cela, nous analysons les variations des fonctions $t_1 \to \underline{b}_{RS}(i, t_1, t_2)$ et $t_2 \to \underline{b}_{RS}(i, t_1, t_2)$. Nous allons montrer comment cela a été fait pour $t_1 \to \underline{b}_{RS}(i, t_1, t_2)$ et dans le cas où $W_i \geq f_i(r_i^{min})(let_i - est_i)$, les autres cas étant déduits de manière similaire. Le tableau 3.2 présente les résultats dans ces autres cas.

**Théorème 3.5.** *Pour chaque activité $i$ et pour tout début d'intervalle $t_1$, il existe au plus deux intervalles $[t_1, t_2[$ tels que l'équation (3.28) soit vérifiée. De même, pour chaque activité $i$ et pour toute fin d'intervalle $t_2$, il existe au plus deux intervalles $[t_1, t_2[$ tels que l'équation (3.27) soit vérifiée. Ces intervalles sont décrits par le tableau 3.2.*

*Démonstration.* Nous présentons seulement comment nous obtenons les $t_2$ pertinents dans le cas où $W_i \geq f_i(r_i^{max})(let_i - est_i)$, i.e. les deux dernières colonnes du tableau 3.2a. Les autres cas sont obtenus de manière similaire. Pour prouver le lemme, nous analysons les variations de la fonction $t_2 \to \underline{b}_{RS}(i, t_1, t_2)$.

| $r_i^{min} = 0$ | $W_i \leq f_i(r_i^{min})(let_i - est_i)$ | | $W_i \geq f_i(r_i^{min})(let_i - est_i)$ | |
|---|---|---|---|---|
| $t_1 < e_i^{min}$ | $t_1 < let_i$ | $s_i^{max} > t_1 > let_i - W_i/f_i(r_i^{min})$ | $t_1 < let_i$ | $s_i^{max} > t_1 > est_i$ |
| $[t_1, let_i[$ | $[t_1, let_i[$ | $[t_1, \Delta(t_1)[$ | $[t_1, let_i[$ | $[t_1, \Delta(t_1)[$ |

(a) $t_2$ pour le placement à droite

| $r_i^{min} = 0$ | $W_i \leq f_i(r_i^{min})(let_i - est_i)$ | | $W_i \geq f_i(r_i^{min})(let_i - est_i)$ | |
|---|---|---|---|---|
| $t_2 > s_i^{max}$ | $t_2 \geq let_i$ | $let_i > t_2 > s_i^{max}$ | $t_2 \geq D_{t_2}$ | $D_{t_2} > t_2 > s_i^{max}$ |
| $[est_i, t_2[$ | $[est_i, t_2[$ | $[\Delta'(t_2), t_2[$ | $[est_i, t_2[$ | $[\Delta'(t_2), t_2[$ |

(b) $t_1$ pour le placement à droite

TABLE 3.2 – Intervalles d'intérêt pour les ajustements du raisonnement énergétique pour le CECSP(placement à droite).

Les intervalles $[t_1, t_2[$ pour lesquels $t_2 \to \underline{b}_{RS}(i, t_1, t_2)$ vérifie que sa dérivée à gauche est supérieure à sa dérivée à droite sont :

- $[t_1, let_i[$ dans le cas où $t_1 \leq est_i$ ;
- $[t_1, let_i[$ et $[t_1, \Delta(t_1)[$ dans le cas où $est_i < t_1 < s_i^{max}$ ;
- $[t_1, let_i[$ dans le cas où $s_i^{max} \leq t_1 < let_i$.

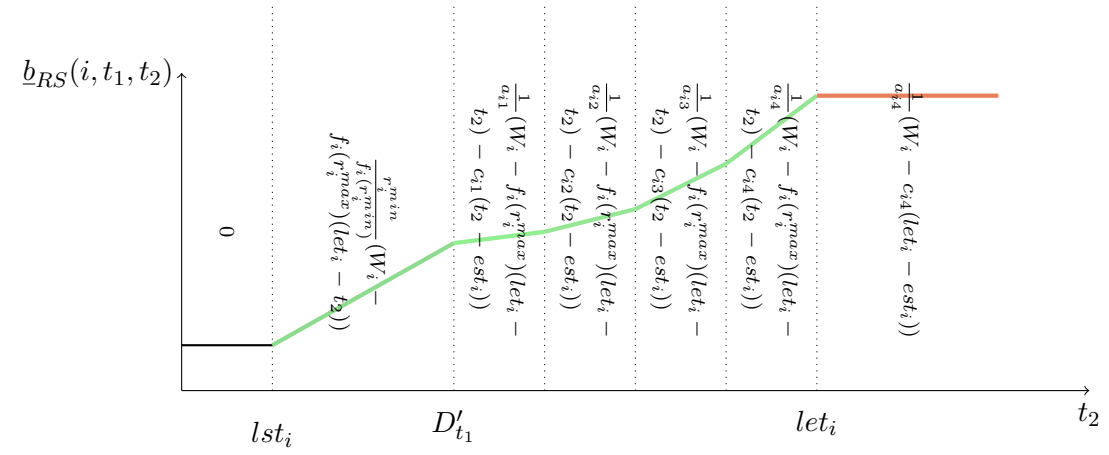
En simplifiant et rassemblant les cas ci-dessus, nous obtenons :

- $[t_1, let_i[$ dans le cas où $t_1 < let_i$ ;
- $[t_1, \Delta(t_1)[$ dans le cas où $est_i < t_1 < s_i^{max}$.

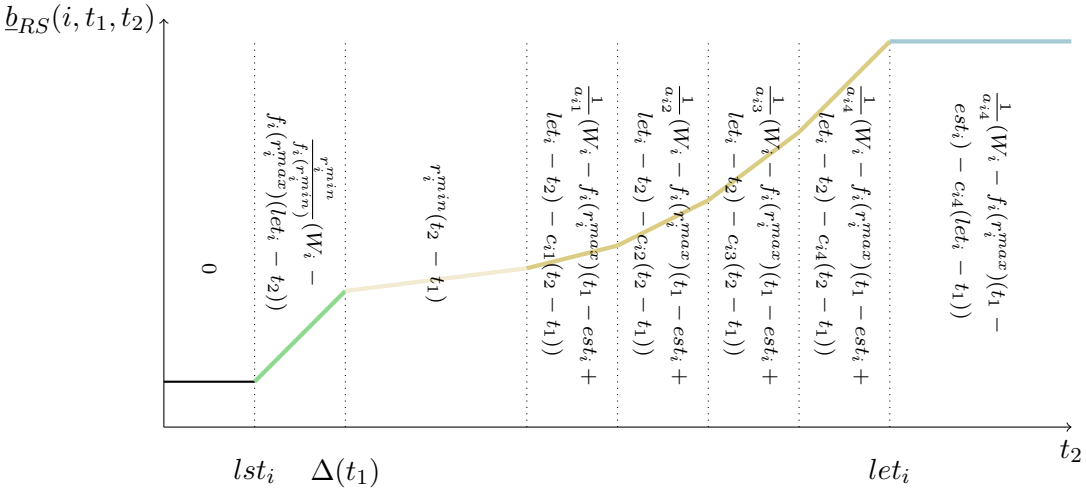
□

Un raisonnement identique amène à la caractérisation des intervalles d'intérêt pour $\underline{b}_{LS}(i, t_1, t_2)$. Ces résultats sont décrits dans le tableau 3.3.

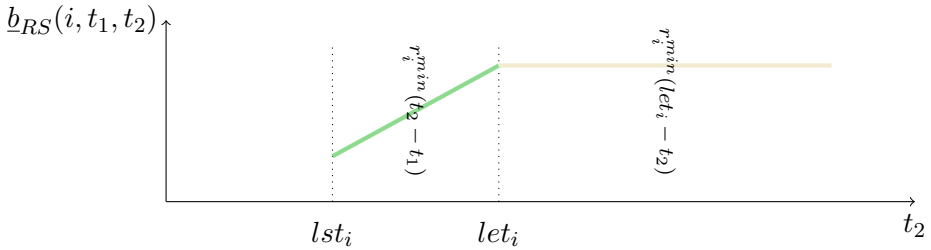
Comme dans le cas de l'algorithme de vérification, cette caractérisation nous permet de définir une liste d'intervalles sur lesquels appliquer les ajustements. Pour cela, il suffit, pour chaque couple $(i, t_1)$ et pour chaque activité $j$, de calculer les $t_2$ d'intérêt à l'aide du tableau 3.2 pour les ajustements des dates de début et à l'aide du tableau 3.3 pour les ajustements des dates de fin. Pour une activité $i$, le nombre d'intervalles d'intérêt est donc au plus de $4(n-1)^2 + 2 * (2n+3)$.




(a) Cas $t_1 \leq est_i$




(b) Cas $est_i < t_1 < s_i^{max}$




(c) Cas $s_i^{max} \leq t_1 < let_i$

FIGURE 3.14 – Variation de $\underline{b}_{RS}(i,t_1,t_2)$ en fonction de $t_2$.

## Conclusion

Dans ce chapitre, nous avons présenté plusieurs adaptations de raisonnements existants ainsi que de nouveaux raisonnements dans le cadre du CECSP. Dans un premier temps, nous avons montré que certains des raisonnements pour la contrainte cumulative pouvaient être directement adaptés

| $r_i^{min} = 0$ | $W_i \leq f_i(r_i^{min})(let_i - est_i)$ | | $W_i \geq f_i(r_i^{min})(let_i - est_i)$ | |
|---|---|---|---|---|
| $t_1 < e_i^{min}$ | $t_1 \leq est_i$ | $e_i^{min} > t_1 > est_i$ | $D'_{t_1} \not\models t_1$ | $e_i^{min} > t_1 > D'_{t_1}$ |
| $[t_1, let_i[$ | $[t_1, let_i[$ | $[t_1, \Gamma(t_1)[$ | $[t_1, let_i[$ | $[t_1, \Gamma(t_1)[$ |

(a) $t_2$ pour le placement à gauche

| $r_i^{min} = 0$ | $W_i \leq f_i(r_i^{min})(let_i - est_i)$ | | $W_i \geq f_i(r_i^{min})(let_i - est_i)$ | |
|---|---|---|---|---|
| $t_2 > s_i^{max}$ | $t_2 > est_i$ | $e_i^{min} < t_2 < est_i + W_i/f_i(r_i^{min})$ | $t_2 > est_i$ | $let_i > t_2 > e_i^{min}$ |
| $[est_i, t_2[$ | $[est_i, t_2[$ | $[\Gamma'(t_2), t_2[$ | $[est_i, t_2[$ | $[\Gamma'(t_2), t_2[$ |

(b) $t_1$ pour le placement à gauche

TABLE 3.3 – Intervalles d'intérêt pour les ajustements du raisonnement énergétique pour le CECSP(placement à gauche).

au CECSP. En effet, dans le CECSP une activité peut être vue comme deux sous-activités, une rectangulaire et une représentant la partie malléable de l'activité. Considérer seulement la partie rectangulaire de l'activité dans le pire des cas, en temps ou en consommation de ressource, permet souvent l'adaptation directe de plusieurs raisonnements. Pour illustrer ce propos, nous avons présenté l'adaptation des raisonnements Time-Table, disjonctif et Time-Table disjonctif. Cette technique nous a aussi permis de développer un nouvel algorithme de vérification basée sur les flots et utilisant la programmation linéaire. Cet algorithme peut être couplé avec d'autres algorithmes, d'ajustement ou de vérification, comme le Time-Table, disjonctif ou non, ou le raisonnement énergétique.

Cependant, d'autres raisonnements ne peuvent être adaptés directement. C'est le cas du raisonnement énergétique pour lequel nous avons présenté l'adaptation au CECSP de l'algorithme de vérification ainsi que des ajustements des fenêtres de temps. De plus, une caractérisation totale des intervalles à considérer dans le cas de l'algorithme de vérification et des ajustements a été détaillée. Dans le cadre de l'algorithme de vérification, plusieurs méthodes permettant de calculer ces intervalles ont été présentées. Les comparaisons de ces différentes méthodes ainsi que les performances des algorithmes de filtrage seront présentées dans le chapitre 6.

# Conclusion

Dans cette partie présentant des méthodes issues de la programmation par contraintes, nous avons commencé par brièvement rappeler les principales notions sur lesquelles reposent ce paradigme. De plus, une description des principales techniques de modélisation des problèmes d'ordonnancement à été donnée.

Ces deux sous-sections introductives nous ont ensuite permis de décrire un problème d'ordonnancement particulier, le problème cumulatif. Pour ce problème, un grand nombre d'algorithmes de vérification et de filtrage ont été présentés. Ces algorithmes se classent en trois catégories : les algorithmes impliquant des raisonnements simples, ceux utilisant le concept d'énergie et enfin ceux combinant plusieurs raisonnements, souvent simples, pour améliorer leurs performances. Pour chacune de ces catégories, au moins un exemple d'un tel algorithme est détaillé : le Time-Table et le raisonnement disjonctif dans le cas des algorithmes simples, le raisonnement énergétique pour les algorithmes utilisant le concept d'énergie et enfin le Time-Table disjonctif dans le dernier cas. Pour le raisonnement énergétique, une étude de Derrien *et al.* [DP14] permettant de caractériser les intervalles d'intérêt du raisonnement énergétique pour la contrainte cumulative est présentée.

Les raisonnements présentés pour la contrainte cumulative sont ensuite adaptés dans le cas du CECSP. Le premier raisonnement que nous avons adapté au CECSP est le raisonnement énergétique. En effet, l'adaptation d'un raisonnement pour la contrainte cumulative dans le cas du CECSP est généralement beaucoup moins fort. De ce fait, nous avons choisi de commencer par adapter un des raisonnements les plus forts au CECSP. Pour ce raisonnement, nous avons aussi étendu au cas du CECSP les travaux de Derrien *et al.* [DP14] pour la contrainte cumulative, ce qui a permis de prouver la polynomialité de ce raisonnement dans le cadre du CECSP. De plus, deux autres méthodes de calcul de ces intervalles pour l'algorithme de vérification sont présentées. Ces deux méthodes reposent sur une étude des segments de définition des fonctions de consommation individuelle des activités.

D'autres adaptations des raisonnements cumulatifs sont aussi détaillés. Ces raisonnements s'adaptent aisément au cas du CECSP. Ils restent cependant beaucoup moins puissants en termes de déduction que leurs homologues cumulatifs. De ce fait, un nouvel algorithme de vérification couplant un programme linéaire adapté d'un problème de flot et le raisonnement Time-Table est présenté. Ce raisonnement n'étant pas dominé par le raisonnement énergétique, il peut être couplé avec ce dernier afin de déduire plus d'incohérence. Il peut aussi être couplé avec des algorithmes de filtrage tels que celui du raisonnement disjonctif ou du Time-Table.

Pour la poursuite de ces travaux, il est raisonnable de chercher à déterminer des règles ou propriétés selon lesquelles un raisonnement a plus de chance d'appliquer des ajustements qu'un autre. De plus, l'adaptation d'autres raisonnements définis pour la contrainte cumulative peut être envisagée.

Cependant, la mise en place de raisonnements dédiés semble être une perspective plus prometteuse.

# Troisième partie

# Programmation linéaire en nombres entiers


**4 Programmation linéaire et ordonnancement de projet** **105**

4.1 La programmation linéaire en nombres entiers . . . . . 105

4.1.1 La programmation linéaire . . . . . 105

4.1.2 Contrainte d'intégrité . . . . . 106

4.1.3 Techniques de résolution . . . . . 106

4.2 Application au RCPSP . . . . . 107

4.2.1 Modèles indexés par le temps . . . . . 108

4.2.2 Modèles à événements . . . . . 109

**5 Programmation linéaire pour le CECSP et renforcement des modèles** **115**

5.1 Modèles de programmation linéaire mixte pour le CECSP . . . . . 115

5.1.1 Modèle indexé par le temps . . . . . 115

5.1.2 Modèles à événements . . . . . 119

5.2 Renforcement des modèles . . . . . 126

5.2.1 Amélioration des modèles indexés par le temps . . . . . 127

5.2.2 Amélioration des modèles à événements . . . . . 129

*Dans cette partie consacrée à l'adaptation des techniques de résolution issues de la programmation linéaire mixte et en nombres entiers, les concepts généraux qu'elles utilisent sont décrits dans le chapitre 4. Ces techniques sont ensuite présentées dans le cadre d'un célèbre problème d'ordonnancement : le RCPSP.*

*Pour ce problème, trois modèles sont présentés. Le premier est un modèle indexé par le temps et fait partie des classes de modèles les plus performants pour la résolution des instances de la PSPLIB [KS96] pour le RCPSP. Cependant, ces modèles ont leurs limitations puisqu'entre autres, leur taille dépent directement de l'horizon de temps du projet. Pour des planifications à long terme, d'autres modèles plus performants sont introduits : les modèles à événements. Ils ont l'avantage d'avoir un nombre polynomial de variables et de contraintes et permettent la modélisation des problèmes continus. Cependant, leurs relaxations linéaires sont beaucoup moins fortes que celles des modèles indexés par le temps, ce qui les rend non compétitifs dans le cas d'instances ayant un petit horizon de temps. Ces modèles dans le cadre du RCPSP ont été introduits par Koné et al. [KALM11].*

*Dans le chapitre 5, nous présentons l'adaptation de ces modèles dans le cadre du CECSP. Ces modèles sont présentés dans [NAL15b, NALR16, NAL15a]. De plus, pour chacun des trois modèles, des techniques visant à améliorer leurs performances sont présentées. Pour le modèle indexé par le temps, nous utilisons le raisonnement énergétique afin de décrire des inégalités valides pouvant être ajoutées au modèle (avant ou pendant la résolution). Cette technique a été présentée dans [NAL16].*

*Pour les modèles à événements, plusieurs ensembles d'inégalités sont présentés : des inégalités permettant de borner la distance entre deux événements, d'autres permettant de borner la date des événements ou des inégalités dérivées du problème du sac-à-dos. De plus, des coupes appelées inégalités de non-préemption sont définies et une étude polyédrale de l'ensemble des vecteurs binaires solution d'un des modèles présenté est détaillée. Ces résultats ont été mis en place en collaboration avec Tamás Kis et ont été présentés dans [NAKL16a, NAKL16b, NKAL16].*

## Chapitre 4

# Programmation linéaire et ordonnancement de projet

### 4.1 La programmation linéaire en nombres entiers

Dans cette section, nous présentons les concepts de base de la programmation linéaire ainsi que les notions et outils qui seront utilisés dans la suite de cette partie. Cette présentation n'étant que partielle, nous renvoyons le lecteur à [Wol98] pour une description plus détaillée des différentes techniques existantes.

#### 4.1.1 La programmation linéaire

La programmation linéaire vise à résoudre des problèmes d'optimisation ayant la particularité de pouvoir s'exprimer à l'aide de contraintes, se présentant sous la forme d'inégalités et/ou d'égalités, linéaires en fonction des variables du problème. De plus, la fonction objectif du problème, i.e. ce que l'on cherche à maximiser/minimiser, s'écrit aussi sous la forme d'une fonction linéaire.

Sous forme canonique, un programme linéaire (PL) s'écrit de la manière suivante :

$$\begin{array}{ll} \text{maximiser} & cx \\ \text{tel que} & Ax \leq b \\ & x \in \mathbb{R}^n_+ \end{array}$$

avec $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ et $A \in \mathbb{R}^{m,n}$.

Nous pouvons supposer, sans perte de généralité, que les variables du problème sont positives et que la fonction objectif $x \to cx$ doit être maximisée. Les vecteurs $x$ de $\mathbb{R}^n$ qui vérifient $Ax \leq b$ sont appelés les solutions réalisables du problème et les vecteurs $x^*$ qui, en plus, maximisent le critère $cx^*$ sont appelés solutions optimales.

La force de ces formulations repose sur le fait que l'ensemble des contraintes du problème définit alors un polyèdre convexe nommé polyèdre des solutions réalisables. De plus, si ce polyèdre est non vide, alors toute solution optimale se trouve forcément sur un sommet, appelé point extrême, de celui-ci. Les solutions optimales peuvent aussi se trouver sur une face du polyèdre, i.e. l'intersection de celui-ci avec un plan défini par une des contraintes du programme linéaire $\{x \in \mathbb{R}^n \mid A_j x = b_j\}$.

Ceci a permis la mise en place de plusieurs algorithmes pour résoudre ces programmes linéaires. Un des plus efficaces en pratique et donc des plus utilisés est l'algorithme du Simplexe. Le principe de

cet algorithme est le parcours "intelligent" des points extrêmes du polyèdre des solutions admissibles : à chaque itération, l'algorithme se "déplace" vers un sommet adjacent de meilleur (ou même) coût. L'algorithme se termine donc lorsque tous les sommets adjacents possèdent un coût moins bon que le sommet courant.

Un des inconvénients de cet algorithme est qu'il a, dans le pire des cas, une complexité exponentielle. D'autres algorithmes permettent de résoudre un programme linéaire en temps polynomial. Cependant, en pratique, l'algorithme du simplexe reste plus efficace et c'est souvent ce dernier qui est implémenté dans les solveurs d'optimisation linéaire.

### 4.1.2 Contrainte d'intégrité

De nombreux problèmes d'optimisation combinatoire ne peuvent s'écrire sous la forme d'un programme linéaire. En effet, pour des problèmes tels que les problèmes d'ordonnancement, tout ou partie des variables doivent être restreintes à prendre des valeurs entières.

Quand toutes les variables du problème sont contraintes à prendre des valeurs entières, on parle de programme linéaire en nombres entiers (PLNE). Lorsque seulement une partie des variables sont autorisées à prendre des valeurs réelles, on parle de programme linéaire mixte (PLM).

De manière générale, un programme linéaire mixte peut s'écrire de la façon suivante :

$$
\begin{aligned}
\text{maximiser} \quad & \sum_{i=1}^{p} c_i x_i + \sum_{j=1}^{n-p} f_j y_j \\
\text{tel que} \quad & \sum_{i=1}^{p} a_{ki} x_i + \sum_{j=1}^{n-p} d_{kj} y_j \leq b_k, \quad k = 1, \ldots, m \\
& x_i \geq 0, \quad i = 1, \ldots, p \\
& y_j \in \mathbb{N}, \quad j = 1, \ldots, n-p
\end{aligned}
$$

$y_j,\ j = 1, \ldots, n-p$ sont les variables de décisions ne pouvant prendre que des valeurs entières tandis que les variables $x_i,\ i = 1, \ldots, p$ peuvent prendre n'importe quelles valeurs réelles positives.

La difficulté de la résolution de ces programmes repose sur le fait que l'ensemble des solutions réalisables du problème ne forme plus un polyèdre convexe et les algorithmes traditionnels ne peuvent être appliqués directement dans ce cas. Le problème de décision associé à un PLNE, ou à un PLM, est NP-complet [GJ79]. De ce fait, plusieurs techniques permettant de trouver la solution optimale en temps raisonnable ont été développées.

### 4.1.3 Techniques de résolution

Parmi les techniques les plus utilisées, on retrouve les techniques utilisant des algorithmes de recherche arborescente par séparation et évaluation ainsi que des méthodes de coupes.

Le principe de la *recherche arborescente par séparation et évaluation* repose sur deux idées principales :

— **la séparation** qui consiste à décomposer selon une partition l'ensemble des solutions en plusieurs sous-ensembles ;

— **l'évaluation** qui consiste à examiner la qualité des solutions d'un sous-ensemble de façon optimiste, i.e. trouver une borne supérieure (resp. inférieure) de la meilleure solution de ce sous-ensemble lorsqu'il s'agit d'un problème de maximisation (resp. minimisation).

L'algorithme propose de parcourir l'arborescence des solutions possibles en évaluant chaque sous-ensemble de solutions de façon optimiste. Lors de ce parcours, il maintient la valeur de la meilleure solution trouvée jusqu'à présent. Quand l'évaluation d'un sous-ensemble donne une valeur plus faible (plus forte pour un problème de minimisation) ou égale, il est inutile d'explorer plus loin ce sous-ensemble. L'ensemble des solutions admissibles est ainsi représenté par une arborescence dans laquelle un grand nombre de nœuds peuvent être éliminés. L'évaluation des sous-problèmes se fait typiquement par *relaxation linéaire* (on ignore la contrainte d'intégrité) en utilisant le Simplexe.

Les *méthodes de coupes* cherchent à caractériser le plus petit polyèdre contenant l'ensemble de toutes les solutions admissibles. Pour cela, un ensemble de coupes, i.e. des inégalités permettant de supprimer une partie du polyèdre ne contenant que des solutions non entières, est généré. Si ces coupes sont assez fortes, le polyèdre des solutions admissibles du modèle devient alors exactement l'enveloppe convexe des solutions admissibles entières. Dans ce cas, la résolution de la relaxation du programme linéaire réduit à cet ensemble donne une solution optimale entière.

La sous-section suivante est dédiée aux modèles de programmation linéaire mis en place pour le problème d'ordonnancement de projet sous contraintes de ressource.

## 4.2 Application à l'ordonnancement de projet sous contraintes de ressources

La facilité de modélisation des problèmes d'ordonnancement sous contraintes de ressources à l'aide de la programmation linéaire en a fait une des principales méthodes de résolution pour ces derniers. De ce fait, de nombreuses techniques de modélisation ont été développées et la littérature sur ce sujet est très vaste. Dans cette section, nous nous intéressons principalement aux modèles développés pour le problème d'ordonnancement de projet sous contraintes de ressources, le RCPSP. Ces modèles sont regroupés en plusieurs familles :

— les formulations indexées par le temps (ou à temps discret) sont, en général, des formulations purement discrètes. Ces formulations ont la particularité d'avoir les meilleures relaxations linéaires au détriment du nombre de variables [CAVT87b, PWW69] ;

— les formulations avec variables de séquencement sont des formulations qui ont l'avantage d'être plus compactes que celles indexées par le temps mais possèdent de moins bonnes relaxations linéaires que ces dernières [OG93, AMR03] ;

— les formulations à événements, plus récentes, possèdent aussi de faibles relaxations linéaires mais ont un nombre polynomial de variables, moins élevé que les formulations à temps discret [KALM11].

Dans cette section, nous présentons quelques modèles de programmation linéaire en nombres entiers ou mixte développés pour les problèmes d'ordonnancement sous contraintes de ressources. Nous nous intéressons particulièrement aux modèles décrits dans [KALM11] pour le RCPSP. En effet, ce sont ces modèles qui seront adaptés dans le cas du CECSP. Nous commençons donc, dans un premier temps, par présenter les modèles indexés par le temps puis, dans un second temps, nous présentons les modèles à événements. Les modèles avec variables de séquencement ne sont pas traités ici. Des exemples de telles formulations dans le cadre du RCPSP sont décrites dans [ADN07].

### 4.2.1 Modèles indexés par le temps

Une des méthodes permettant la modélisation des problèmes d'ordonnancement de projet consiste à discrétiser l'horizon de temps. Ces modèles de programmation linéaire en nombres entiers indexés par le temps, aussi appelés modèles à temps discret, ont été largement étudiés dans le cadre des problèmes d'ordonnancement en général. Ceci étant principalement dû à leur relativement forte relaxation linéaire et à la facilité d'extension à de nombreux objectifs et contraintes.

Dans ces formulations, l'horizon de temps est discrétisé en périodes de temps, i.e en intervalles, généralement unitaires, et un ensemble de variables est défini pour modéliser le statut d'une activité $i$ en période $t$, en cours, commencé ou terminé. Pour chaque activité $i$ et pour chaque période $t$ (discret), une variable $x_{it} \in \{0, 1\}$ est donc définie.

Dans la formulation présentée pour le RCPSP, cette variable prendra la valeur 1 si et seulement si l'activité $i$ commence au début de la période $t$. Le nombre de ces variables dépend donc de la taille de l'horizon de temps du problème, i.e. du nombre de périodes, qui peut être, pour certains problèmes, aussi grand que la somme de toutes les durées des activités. Le calcul d'une borne supérieure raisonnable pour la durée du projet $T$ est donc indispensable. Dans la suite, nous noterons $\mathcal{H} = \{0, \ldots, T-1\}$ l'ensemble des périodes du problème.

En pratique, pour chaque activité, nous calculons l'ensemble des périodes de temps pendant lesquels l'activité peut commencer. Pour cela, nous ajoutons deux activités fictives au problème : 0 et $n+1$. Ces activités ont les caractéristiques suivantes : leur durée est égale à 0, elles ne consomment aucune ressource durant leur exécution et l'activité 0 (resp. $n+1$) doit être ordonnancée avant (resp. après) toutes les autres activités. Ces deux activités fictives vont nous permettre d'associer à chaque activité $i$, une date de début au plus tôt $est_i$ et une date de début au plus tard $lst_i$. Donc, l'intervalle $[est_i, lst_i]$ est la fenêtre de temps pendant laquelle l'activité $i$ peut commencer.

Pour calculer ces fenêtres de temps, remarquons que, si chaque arc $(i, j)$ du graphe des précédences $G$ est pondéré par $p_i$, la date de début au plus tôt de $i$, $est_i$ peut prendre la valeur du plus long chemin entre l'activité 0 et l'activité $i$ et la date de début au plus tard $lst_i$, $T$ moins la valeur du plus long chemin entre $i$ et $n+1$.

Dans la suite, nous notons $\mathcal{A}$ l'ensemble des activités réelles du projet et $V = \mathcal{A} \cup \{0, n+1\}$ l'ensemble des activités réelles et fictives du projet.

Nous pouvons maintenant exhiber la formulation à temps discret suivante [PWW69] :

$$\min \sum_{t \in \mathcal{H}} t x_{n+1,t} \tag{4.1}$$

$$\sum_{t \in \mathcal{H}} t x_{jt} - \sum_{t \in \mathcal{H}} t x_{it} \geq p_i \qquad \forall (i,j) \in E \tag{4.2}$$

$$\sum_{i \in V} \sum_{\tau = \max(0, t-p_i+1)}^{t} r_{ik} x_{i\tau} \leq R_k \qquad \forall t \in \mathcal{H},\ \forall k \in \mathcal{R} \tag{4.3}$$

$$\sum_{t \in \mathcal{H}} x_{it} = 1 \qquad \forall i \in V \tag{4.4}$$

$$x_{it} \in \{0, 1\} \qquad \forall i \in V,\ \forall t \in \mathcal{H} \tag{4.5}$$

Dans cette formulation, l'objectif est donné par l'expression (4.1). Cette expression signifie que l'on cherche à minimiser la date de début de l'activité $n+1$. Or comme cette activité est forcément placée à la fin de l'ordonnancement du fait des relations de précédences, ceci revient bien à minimiser la durée totale du projet.

La contrainte (4.2) modélise les relations de précédences entre les activités. En effet, toute paire d'activités $(i, j)$ ayant la propriété que $i$ doit précéder $j$ vérifie la relation suivante : la date de début de $j$ est supérieure ou égale à la date de début de $i$ plus sa durée, i.e. la date de fin de $i$.

La contrainte (4.3) formalise les contraintes de ressource. En effet, la contrainte s'assure que, pour chaque ressource $k \in \mathcal{R}$, la somme des consommations instantanées sur $k$ des activités en cours durant la période $t$ est bien inférieure ou égale à la capacité $R_k$ de cette ressource.

La contrainte (4.4) impose que chaque activité n'ait qu'une et une seule date de début. Ceci peut aussi être vu comme une contrainte de non-préemption puisque ceci revient à empêcher que l'activité soit interrompue et redémarrée plus tard dans l'ordonnancement final.

Le modèle possède donc $(n+2)T$ variables binaires et $|E| + T * m + n$ contraintes.

Il existe d'autres formulations à temps discret comme la formulation désagrégée de Christofides et al. [CAVT87b] ou la formulation basée sur les ensembles réalisables de Mingozzi et al. [MMRB98]. Ces formulations ne seront pas détaillées dans ce manuscrit. Une description des modèles de PLNE existants pour le RCPSP peut être trouvée dans [ADN07].

### 4.2.2 Modèles à événements

Nous allons présenter deux formulations à événements : une formulation dite start/end et une formulation dite on/off. Cette sous-section montre, dans un premier temps, l'intérêt de l'utilisation des modèles à événements à la place des modèles à temps discret, en comparant notamment leur nombre respectif de variables et de contraintes. Dans un second temps, les modèles seront détaillés.

#### Motivations des modèles à événements

Parmi les formulations existantes pour le RCPSP, celles basées sur des modèles indexés par le temps sont les plus utilisées dû au fait de la qualité (relative) de leur relaxation. Cependant, comme la taille de ces modèles, et donc leur complexité, dépend grandement de la taille de l'horizon de temps du problème, ces modèles peuvent s'avérer moins efficaces sur certains types d'instances [KALM11]. Pour pallier ce problème, des formulations basées sur des *événements* ont été mises en place.

La notion d'événement permet de caractériser seulement les dates "importantes" de l'ordonnancement, i.e. les dates de début et de fin de chaque activité. Ainsi, la considération de chaque date de l'horizon de temps n'est plus nécessaire. Cela permet de réduire le nombre de variables qui ne dépend alors que du nombre d'activités à ordonnancer. De plus, pour le RCPSP, seuls les événements correspondant au début d'une activité ont besoin d'être considérés. En effet, si l'on considère que les activités sont ordonnancées au plus tôt, i.e. dès que les ressources requises sont disponibles, alors la date de début de chaque activité correspond soit à l'instant 0, soit à la date de fin d'une tâche.

Les formulations à événements possèdent aussi la caractéristique suivante : elles permettent de

résoudre des instances pouvant contenir des valeurs (de paramètre et/ou de solution) non entières.

Dans [KALM11], les auteurs montrent les limitations des modèles indexés par le temps pour le RCPSP. Les instances considérées sont des instances pour lesquelles l'horizon de temps, i.e. $T$, est très grand. Dans ce cas-là, le nombre de variables et de contraintes des modèles à temps discret est nettement supérieur à ceux des modèles à événements.

Par exemple, la famille d'instances KSD15_d [Kon09] possède 480 instances de 15 activités et avec 4 ressources dont l'horizon de temps peut varier entre 187 et 999 et possédant entre 77 et 144 relations de précédence. Pour le RCPSP, le modèle à temps discret peut alors posséder jusqu'à 17000 variables binaires et 4200 contraintes. Les modèles à événements possèdent quant à eux jusqu'à 500 variables binaires, 16 variables continues et 4200 contraintes pour le modèle start/end et jusqu'à 250 variables binaires, 16 variables continues et 4000 contraintes.

Considérant des instances de taille similaire pour le CECSP avec fonction de rendement affine, le modèle à temps discret peut alors posséder jusqu'à 30000 variables binaires, le même nombre de variables continues et 90000 contraintes. Les modèles à événements possèdent quant à eux jusqu'à 900 variables binaires, le même nombre de variables continues et 7200 contraintes pour le modèle start/end et jusqu'à 450 variables binaires, 900 variables continues et 7100 contraintes.

Dans ce cas-là, la faiblesse de la relaxation linéaire des modèles à événements est largement compensée par le grand nombre de variable et de contraintes des modèles indexés par le temps.

Dans ce contexte, l'amélioration des modèles à événements s'avère être une direction de recherche nécessaire dans l'élaboration de méthodes de résolution efficaces pour le RCPSP.

### Modèle start/end

Dans cette formulation, la date de chaque événement est représentée par une variable continue $t_e$, pour tout $e \in \mathcal{E}$, avec $\mathcal{E}$ l'ensemble des indices des événements. Afin d'associer ces dates aux débuts et fins des activités, nous définissons, pour toute activité $i \in \mathcal{A}$ et pour tout événement $e \in \mathcal{E}$, deux variables binaires, $x_{ie}$ et $y_{ie}$, ayant les propriétés suivantes :

- $x_{ie} = 1$ si et seulement si l'activité $i$ commence à l'événement $e$, c'est-à-dire commence à la date $t_e$.
- $y_{ie} = 1$ si et seulement si l'activité $i$ finit à l'événement $e$, c'est-à-dire finit à la date $t_e$.

Notons que les événements correspondant à la fin d'une activité correspondent aussi au début d'une autre activité (à l'exception de l'événement correspondant à la date de fin de la dernière activité), le nombre d'événements à considérer est $n + 1$. De ce fait, nous avons comme ensemble d'événements $\mathcal{E} = \{1, \ldots, n + 1\}$.

Nous définissons aussi une variable continue $b_{ek}$, pour chaque événement $e \in \mathcal{E}$ et pour chaque ressource $k \in \mathcal{R}$, modélisant la consommation totale de la ressource $k$ à l'événement $e$ (et donc durant tout l'intervalle compris entre $t_e$ et $t_{e+1}$). Ceci nous permet de présenter la formulation suivante issue de [KALM11] et corrigées dans [ABK+13] :

$$\min t_{n+1} \tag{4.6}$$

$$t_1 = 0 \tag{4.7}$$

$$t_e \leq t_{e+1} \qquad \forall e \in \mathcal{E} \tag{4.8}$$

$$\sum_{e\in\mathcal{E}} x_{ie} = 1 \qquad \forall i \in \mathcal{A} \tag{4.9}$$

$$\sum_{e\in\mathcal{E}} y_{ie} = 1 \qquad \forall i \in \mathcal{A} \tag{4.10}$$

$$est_i x_{ie} \leq t_e \leq lst_i x_{ie} + lst_{n+1}\,(1 - x_{ie}) \qquad \forall i \in \mathcal{A},\ \forall e \in \mathcal{E} \tag{4.11}$$

$$(lst_i + p_i)\, y_{ie} + (1 - y_{ie})\, lst_{n+1} \geq t_e \qquad \forall i \in \mathcal{A},\ \forall e \in \mathcal{E} \tag{4.12}$$

$$t_e \geq y_{ie}\,(est_i + p_i) \qquad \forall i \in \mathcal{A},\ \forall e \in \mathcal{E} \tag{4.13}$$

$$b_{1k} - \sum_{i\in\mathcal{A}} r_{ik} x_{i1} = 0 \qquad \forall k \in \mathcal{R} \tag{4.14}$$

$$b_{ek} - b_{e-1,k} + \sum_{i\in\mathcal{A}} r_{ik}\,(y_{ie} - x_{ie}) = 0 \qquad \forall e \in \mathcal{E}\setminus\{1\},\ \forall k \in \mathcal{R} \tag{4.15}$$

$$b_{ek} \leq R_k \qquad \forall e \in \mathcal{E},\ \forall k \in \mathcal{R} \tag{4.16}$$

$$t_f \geq t_e + (x_{ie} + y_{if} - 1)p_i \qquad \forall i \in \mathcal{A},\ \forall (e,f) \in \mathcal{E}^2,\ f > e \tag{4.17}$$

$$\sum_{f=1}^{e} y_{if} + \sum_{f=e}^{n} x_{if} \leq 1 \qquad \forall i \in \mathcal{A},\ \forall e \in \mathcal{E} \tag{4.18}$$

$$\sum_{f=e}^{n} y_{if} - \sum_{f=1}^{e-1} x_{jf} \leq 1 \qquad \forall (i,j) \in E,\ \forall e \in \mathcal{E} \tag{4.19}$$

$$est_{n+1} \leq t_n \leq lst_{n+1} \tag{4.20}$$

$$t_e \geq 0 \qquad \forall e \in \mathcal{E} \tag{4.21}$$

$$b_{ek} \geq 0 \qquad \forall k \in \mathcal{R},\ \forall e \in \mathcal{E} \tag{4.22}$$

$$x_{ie} \in \{0,1\},\ y_{ie} \in \{0,1\} \qquad \forall i \in \mathcal{A},\ \forall e \in \mathcal{E} \tag{4.23}$$

Dans cette formulation, l'objectif est donné par l'équation (4.6). En effet, comme pour la formulation à temps discret, on cherche à minimiser la date de fin du projet. Or, l'événement $n+1$ représente, par définition, le dernier événement, c'est à dire le début de l'activité fictive $n+1$ et donc la fin de l'ordonnancment.

La contrainte (4.7) fixe le premier événement à la date 0, tandis que la contrainte (4.8) ordonne les dates des événements par ordre croissant.

La contrainte (4.9) (resp. (4.10)) stipule que chaque activité ne peut commencer (resp. finir) qu'une et une seule fois. En effet, chaque début (resp. fin) d'activité ne peut être associé qu'à un et un seul événement.

La contrainte (4.11) garantit que la date d'un événement correspondant au début d'une activité soit bien comprise dans sa fenêtre de temps, i.e. entre sa date de début au plus tôt $est_i$ et sa date de début au plus tard $lst_i$. En effet, si l'événement $e$ correspond au début de l'activité $i$, i.e. $x_{ie} = 1$, alors l'inégalité devient : $est_i \leq t_e \leq lst_i$. Au contraire, si $x_{ie} = 0$, i.e. $e$ ne correspond pas au début de $i$, alors l'inégalité devient : $0 \leq t_e \leq lst_{n+1}$ et, grâce aux contraintes (4.8) et (4.20), ceci est vrai pour tout $e \in \mathcal{E}$. De même, les contraintes (4.12) et (4.13) s'assurent que, si un événement $f$ correspond à la fin d'une activité $i$, alors $t_f$ est bien compris entre $est_i + p_i$ et $lst_i + p_i$.

Les contraintes (4.14) et (4.15) représentent les contraintes de conservation des ressources. La contrainte (4.14) modélise la consommation totale de chaque ressource $k$ à l'événement 1. Pour une ressource $k$, cette quantité est égale à la somme des consommations sur $k$ de chaque activité commençant à l'événement 1. La contrainte (4.15) donne la consommation totale de chaque ressource $k$

à l'événement $e$, $b_{ek}$. En effet, pour chaque événement $e$, cette quantité est égale à la consommation totale à l'événement $e-1$, i.e. $b_{e-1,k}$, à laquelle on retranche les consommations des activités finissant à l'événement $e$ et on ajoute les consommations de chaque activité commençant à l'événement $e$. Enfin, la contrainte (4.16) limite la demande totale sur chaque ressource à sa capacité.

La contrainte (4.17) assure que si l'activité $i$ commence à l'événement $e$ et se termine à l'événement $f$, alors les dates correspondant à ces deux événements sont séparées par au moins la durée de cette tâche, i.e. $p_i$. Dans ce cas, l'inégalité s'écrit $t_f \geq t_e + p_i$, et pour toute autre combinaison de $x_{ie}$ et $y_{if}$, on obtient soit $t_f \geq t_e$ ou $t_f \geq t_e - p_i$ ce qui est vérifié par la contrainte (4.8).

La contrainte (4.18) garantit qu'une activité ne peut se terminer avant d'avoir commencé. Si l'activité $i$ finit entre l'événement 1 et $e$, i.e. $\sum_{f=1}^{e} y_{if} = 1$, alors elle ne peut pas commencer entre l'événement $e$ et $n$, i.e. $\sum_{f=e}^{n} x_{if} = 0$, et inversement.

Enfin, la contrainte (4.19) modélise les relations de précédences entre les activités : si une activité $i$, devant précéder une activité $j$, finit à l'événement $e$ ou après, alors l'activité $j$ ne peut commencer avant l'événement $e$, i.e. $\sum_{f=e}^{n} x_{if} = 1 \Rightarrow \sum_{f=1}^{e-1} y_{jf} = 0$.

Le modèle possède $2n^2+n$ variables binaires, $n+1$ variables continues et $(n+1)(m+n^2/2+|E|)+3n$ contraintes. Le nombre de variables et de contraintes du modèle est donc bien polynomial en fonction du nombre d'activités et de ressources.

### Modèle on/off

Dans le modèle on/off, comme pour la formulation start/end, la variable $t_e$ représente la date de chaque événement, pour tout $e \in \mathcal{E}$. Pour associer ces dates aux débuts et fins de chaque activité, nous définissons, pour chaque activité $i \in \mathcal{A}$ et pour chaque événement $e \in \mathcal{E}$, la variable $z_{ie} \in \{0,1\}$. Cette variable prendra la valeur 1 si et seulement si l'activité $i$ est en cours durant l'intervalle $[t_e, t_{e+1}]$, et 0 sinon. Ainsi, une activité commencera (resp. finira) à l'événement $e$ si $z_{ie} - z_{i,e-1} = 1$ (resp. $z_{i,e-1} - z_{ie} = 1$).

Notons que, comme dans le cas du modèle start/end, nous pouvons limiter le nombre d'événements au début de chaque activité. Ainsi, $\mathcal{E} = \{1, \ldots, n\}$ ; l'utilisation d'activités fictives n'est, ici, plus nécessaire pour modéliser l'événement de fin de la dernière activité. De plus, la variable $z_{ie}$ modélisant le fait qu'un variable soit en cours entre $t_e$ et $t_{e+1}$, nous pouvons limiter le nombre de ces variables en ne les définissant que pour les événements $e \in \mathcal{E} \setminus \{n\}$. Ceci conduit à la formulation suivante [KALM11] :

$$\min C_{max} \tag{4.24}$$

$$C_{max} \geq t_e + (z_{ie} - z_{i,e-1})p_i \qquad \forall e \in \mathcal{E} \setminus \{1\},\ \forall i \in \mathcal{A} \tag{4.25}$$

$$t_1 = 0 \tag{4.26}$$

$$t_e \leq t_{e+1} \qquad \forall e \in \mathcal{E} \setminus \{n\} \tag{4.27}$$

$$\sum_{e \in \mathcal{E} \setminus \{n\}} z_{ie} \geq 1 \qquad \forall i \in \mathcal{A} \tag{4.28}$$

$$est_i z_{ie} \leq t_e \qquad \forall e \in \mathcal{E} \setminus \{n\},\ \forall i \in \mathcal{A} \tag{4.29}$$

$$t_e \leq lst_i(z_{ie} - z_{i,e-1}) + (1 - (z_{ie} - z_{i,e-1}))lst_{n+1} \qquad \forall e \in \mathcal{E} \setminus \{n\} \setminus \{1\},\ \forall i \in \mathcal{A} \tag{4.30}$$

$$t_f \geq t_e + ((z_{ie} - z_{i,e-1}) - (z_{if} - z_{i,f-1}) - 1)\, p_i \qquad \forall e > f \in (\mathcal{E} \setminus \{1,n\})^2,\ \forall i \in \mathcal{A} \tag{4.31}$$

$$\sum_{f=1}^{e} z_{if} \leq e(1 - (z_{ie} - z_{i,e-1})) \qquad \forall e \in \mathcal{E} \setminus \{1, n\},\ \forall i \in \mathcal{A} \tag{4.32}$$

$$\sum_{f=e}^{n-1} z_{if} \leq (n - e)(1 + (z_{ie} - z_{i,e-1})) \qquad \forall e \in \mathcal{E} \setminus \{1, n\},\ \forall i \in \mathcal{A} \tag{4.33}$$

$$\sum_{i \in \mathcal{A}} r_{ik} z_{ie} \leq R_k \qquad \forall e \in \mathcal{E} \setminus \{n\},\ \forall k \in \mathcal{R} \tag{4.34}$$

$$z_{ie} + \sum_{f=1}^{e} z_{jf} \leq 1 + (1 - z_{ie})e \qquad \forall e \in \mathcal{E} \setminus \{1, n\},\ \forall (i, j) \in E \tag{4.35}$$

L'objectif, donné par l'équation (4.24), consiste à minimiser la date de fin du projet, ici représentée par la variable $C_{max}$. La contrainte (4.25) s'assure que $C_{max}$ prend bien la valeur de la date de fin du projet.

Les contraintes (4.26) et (4.27) jouent le même rôle que dans la formulation start/end, à savoir, ordonner les événements.

La contrainte (4.28) stipule que chaque activité doit être ordonnancée sur au moins un événement dans toute la durée du projet.

Les contraintes (4.29) et (4.30) garantissent que la date d'un événement correspondant au début d'une activité soit bien comprise dans sa fenêtre de temps, i.e. entre sa date de début au plus tôt $est_i$ et sa date de début au plus tard $lst_i$. En effet, si l'événement $e$ correspond au début de l'activité $i$, i.e. $z_{ie} = 1$ et $z_{i,e-1} = 0$ , l'inégalité devient alors : $est_i \leq t_e \leq lst_i$. Nous distinguons 3 autres cas :

- $z_{ie} = z_{i,e-1} = 1$ *:* l'activité est en cours entre les événements $t_{e-1}$ ($z_{i,e-1} = 1$) et $t_{e+1}$ ($z_{ie} = 1$). L'inégalité devient : $est_i \leq t_e \leq lst_{n+1}$. Comme l'activité $i$ a déjà commencé à un événement $f < e$, on a : $est_i \leq t_f \leq t_e$. L'autre inégalité est triviale.
- $z_{ie} = 0$ *et* $z_{i,e-1} = 1$ *:* l'activité se termine à l'événement $e$. L'inégalité donne alors $0 \leq t_e \leq 2 * lst_{n+1} - lst_i$. Or, $2 * lst_{n+1} - lst_i \geq lst_{n+1}$. Donc l'inégalité est vérifiée.
- $z_{ie} = z_{i,e-1} = 0$ *:* l'activité n'est pas en cours entre les événements $t_{e-1}$ et $t_{e+1}$. L'inégalité devient $0 \leq t_e \leq lst_{n+1}$ et est trivialement vérifiée.

La contrainte (4.31) assure une séparation suffisante, i.e. la durée de l'activité, entre un événement correspondant au début d'une activité et un événement correspondant à la fin de cette même activité. La validité de cette contrainte suit la même logique que pour la contrainte (4.17) du modèle précédent.

Les contraintes (4.32) et (4.33), appelées *contraintes de contiguïté*, assure la non-préemption des activités. La validité de cette contrainte n'est pas détaillée ici mais une preuve formelle peut être trouvée dans [KALM11].

La contrainte (4.34) représente les limites de capacité de chaque ressource. En effet, une activité $i$ consomme une quantité $r_{ik}$ de la ressource $k$ entre $t_e$ et $t_{e+1}$ si et seulement si elle est en cours entre ces deux dates, i.e. $z_{ie} = 1$. On pourra remarquer que dans ce modèle le nombre de contraintes nécessaires pour modéliser les contraintes de capacité des ressources est nettement inférieur à celui du modèle précédent.

Enfin, la contrainte (4.35) modélise les relations de précédences entre les activités : si une activité $i$, devant précéder une activité $j$, est en cours entre les événements $e$ et $e+1$, i.e. $z_{ie} = 1$, alors l'activité $j$ ne peut être en cours avant l'événement $e$, i.e. $z_{ie} = 1 \Rightarrow \sum_{f=0}^{e} z_{jf} = 0$.

Le modèle possède $n^2$ variables binaires, deux fois moins que pour le modèle start/end, $n+1$ variables continues et $(n-1)(3+|E|+m+n^2/2)+n^2+n$ contraintes. Le nombre de variables et de contraintes du modèle est donc bien polynomial en fonction du nombre d'activités et de ressources.

### Conclusion

Dans ce chapitre, nous avons rappelé les principales notions de la programmation linéaire, puis de la programmation linéaire mixte et en nombres entiers. Nous avons ensuite vu plusieurs méthodes de modélisation pour un problème d'ordonnancement avec contraintes de ressource : le RCPSP. Pour ce problème, nous avons vu deux types de modèles. Le premier fait partie des modèles indexés par le temps. Ces derniers possèdent de relativement bonnes relaxations, ce qui en fait un des types de modèles les plus efficaces pour le RCPSP et en particulier sur les célèbres instances de la PSPLIB [KS96].

Cependant, ces modèles souffrent de limitations dues à leur taille, dépendante de l'horizon de temps du projet. De ce fait, quand l'horizon de temps des instances devient grand, la taille des modèles associés augmente et leurs performances décroissent. Pour pallier ce problème, un autre type de modèle est mis en place : les modèles à événements. L'avantage de ces modèles est leur taille, polynomiale en fonction de la taille de l'instance et surtout indépendante de l'horizon de temps. Dans [KALM11], Koné *et al.* comparent les performances de ces modèles avec ceux indexés par le temps sur des instances ayant de grands horizons de temps. De plus, ces modèles permettent la modélisation de problèmes continus. De ce fait, nous allons adapter ces modèles dans le cadre du CECSP dans le chapitre suivant. De plus, un modèle indexé par le temps est aussi présenté, permettant d'avoir des solutions plus rapidement (mais pas forcément optimales ou, sans assurance qu'une infaisabilité corresponde à une infaisabilité du problème continu).

## Chapitre 5

# Programmation linéaire pour le CECSP et renforcement des modèles

### 5.1 Modèles de programmation linéaire mixte pour le CECSP

Le CECSP et le RCPSP étant deux problèmes relativement proches – utilisation d'une ou plusieurs ressources cumulatives, de fenêtres de temps – la modélisation du CECSP à l'aide de la programmation linéaire mixte est une approche naturelle pour sa résolution.

De plus, le théorème 1.2 nous assurant que toute solution $S$ du CECSP peut être transformée en une solution $S'$ ayant la propriété que $\forall i \in \mathcal{A},\ b_i(t)$ soit constante par morceaux, le problème peut être modélisé à l'aide de la programmation linéaire mixte. Nous pouvons aussi remarquer que, comme la transformation proposée par le théorème n'augmente ni la date de début des activités, ni leur date de fin, ni leur consommation totale de ressource, tout modèle ayant un objectif impliquant seulement la minimisation de ces trois quantités sera valide pour le CECSP.

Dans cette section, nous commençons par décrire un modèle indexé par le temps, puis, nous présentons deux modèles à événements. Ces trois modèles sont adaptés des modèles pour le RCPSP présentés dans la section 4.2 et sont présentés dans [NALR16].

#### 5.1.1 Modèle indexé par le temps

La première formulation proposée est une formulation indexée par le temps. Ce modèle a été conçu dans le cadre d'une collaboration avec David Rivreau [NALR16]. Comme pour le modèle à temps discret du RCPSP l'horizon de temps est ici divisé en périodes de taille unitaire. Le calcul d'une borne supérieure $T$ sur la durée totale du projet est trivial : il suffit de prendre la plus grande date échue, i.e. $T = \max_{i \in \mathcal{A}} let_i$. L'ensemble des périodes, noté $\mathcal{H}$, peut donc être défini par : $\mathcal{H} = \{0, \ldots, T-1\}$. Notons que, par translation, nous pouvons toujours supposer que $\min_{i \in \mathcal{A}} est_i = 0$.

Une des principales différences entre le modèle à temps discret du RCPSP et celui du CECSP repose sur le fait que, dans le second, la durée des activités n'est pas connue à l'avance et doit être déterminée par le modèle. De ce fait, nous devons définir, pour chaque activité $i \in \mathcal{A}$ et pour chaque instant $t \in \mathcal{H}$, deux variables binaires $x_{it}$ et $y_{it}$ pour modéliser le début et la fin des activités. La variable $x_{it}$ (resp. $y_{it}$) prendra la valeur 1 si et seulement si l'activité $i$ commence (finit) à l'instant $t$.

Une seconde différence entre les deux modèles est le calcul des fenêtres de temps, effectué de

manière triviale pour le CECSP. En effet, pour une activité $i$, la fenêtre correspondante à sa date de début est $[est_i, lst_i]$, avec $lst_i = let_i - W_i/f_i(r_i^{max})$, et celle correspondante à sa date de fin $[eet_i, let_i]$, avec $eet_i = est_i + W_i/f_i(r_i^{max})$. Enfin, l'ajout des activités fictives marquant le début et la fin du projet n'est pas nécessaire ici.

**Fonction de rendement identité** Dans le cas où la fonction de rendement de chaque activité est la fonction identité, i.e. $f_i(b_i(t)) = b_i(t),\ \forall i \in \mathcal{A}$, nous définissons une variable $b_{it}$, pour chaque activité $i \in \mathcal{A}$ et pour chaque période de temps $t \in \mathcal{H}$, qui représentera la quantité de ressource consommée par l'activité $i$ dans la période de temps $t$, i.e. dans l'intervalle $[t, t+1]$.

Ceci conduit à la formulation suivante :

$$\min \sum_{i \in \mathcal{A}} \sum_{t \in \mathcal{H}} b_{it} \tag{5.1}$$

$$\sum_{t=est_i}^{lst_i} x_{it} = 1 \qquad \forall i \in \mathcal{A} \tag{5.2}$$

$$\sum_{t=eet_i}^{let_i} y_{it} = 1 \qquad \forall i \in \mathcal{A} \tag{5.3}$$

$$\left( \sum_{\tau=est_i}^{t} x_{i\tau} - \sum_{\tau=est_i+1}^{t} y_{i\tau} \right) r_i^{min} \leq b_{it} \qquad \forall t \in \{est_i, \dots, let_i - 1\},\ \forall i \in \mathcal{A} \tag{5.4}$$

$$\left( \sum_{\tau=est_i}^{t} x_{i\tau} - \sum_{\tau=est_i+1}^{t} y_{i\tau} \right) r_i^{max} \geq b_{it} \qquad \forall t \in \mathcal{H},\ \forall i \in \mathcal{A} \tag{5.5}$$

$$\sum_{t=est_i}^{let_i} b_{it} \geq W_i \qquad \forall i \in \mathcal{A} \tag{5.6}$$

$$\sum_{i \in \mathcal{A}} b_{it} \leq R \qquad \forall t \in \mathcal{H} \tag{5.7}$$

$$b_{it} = 0 \qquad \forall t \notin \{est_i, \dots, let_i - 1\},\ \forall i \in \mathcal{A} \tag{5.8}$$

$$x_{it} = 0 \qquad \forall t \notin \{est_i, \dots, lst_i\},\ \forall i \in \mathcal{A} \tag{5.9}$$

$$y_{it} = 0 \qquad \forall t \notin \{eet_i, \dots, let_i\},\ \forall i \in \mathcal{A} \tag{5.10}$$

$$b_{it} \geq 0 \qquad \forall t \in \mathcal{H}; \forall i \in \mathcal{A} \tag{5.11}$$

$$x_{it} \in \{0,1\},\ y_{it} \in \{0,1\} \qquad \forall t \in \mathcal{H},\ \forall i \in \mathcal{A} \tag{5.12}$$

Ce modèle s'inspire grandement du modèle de [ALR12], la principale différence résidant dans la considération de fenêtres de temps plus fines pour les variables, i.e. $[est_i, lst_i]$ pour $x_{it}$ au lieu de $[est_i, let_i - 1]$ et $[eet_i, let_i]$ pour $y_{it}$ au lieu de $[est_i - 1, let_i]$. Ces fenêtres de temps pouvant être raffinées à l'aide d'heuristiques, de la programmation par contraintes (cf. sous-section 3.2.2) ou par le biais d'autres méthodes, ces modifications peuvent grandement améliorer les performances du modèle.

Dans cette formulation, l'objectif est décrit par l'équation (5.1). Ici, l'objectif est de minimiser la consommation totale de la ressource durant tout le projet. Cet objectif a moins d'impact dans le cas où la fonction de rendement de chaque activité est la fonction identité mais se révèle très pertinent pour d'autres types de fonctions de rendement. Cependant, comme la formulation à temps discret ne nous permet pas de s'assurer que la quantité de ressource consommée par une activité soit exactement égale à la quantité nécessaire pour apporter l'énergie requise, i.e. $\sum_{t=est_i}^{let_i} b_{it} \geq W_i$, cet objectif reste

valide dans le cas des fonctions rendement identités.

Si l'on souhaite modifier l'objectif pour minimiser la date de fin du projet, il suffit d'introduire une variable $C_{max}$ ainsi que la contrainte suivante :

$$C_{max} \geq \sum_{i \in \mathcal{A}} t y_{it} \quad \forall t \in \mathcal{H}$$

et alors l'objectif s'écrit facilement comme :

$$\min C_{max}$$

Les contraintes (5.2) et (5.3) stipulent que chaque activité est exécutée une et une seule fois durant la durée du projet. En effet, grâce à ces contraintes, chaque activité n'a qu'une et une seule date de début et une et une seule date de fin.

Les contraintes (5.4) et (5.5) permettent de s'assurer que la consommation instantanée de chaque activité est bien comprise entre $r_i^{min}$ et $r_i^{max}$ durant toute sa durée d'exécution. Pour s'en assurer, il suffit de remarquer que $\sum_{\tau=est_i}^{t} x_{i\tau} - \sum_{\tau=est_i+1}^{t} y_{i\tau} = 1$ si et seulement si l'activité $i$ est en cours à l'instant $t$. Les autres configurations possibles sont $\sum_{\tau=est_i}^{t} x_{i\tau} - \sum_{\tau=est_i+1}^{t} y_{i\tau} = 0$ et $\sum_{\tau=est_i}^{t} x_{i\tau} - \sum_{\tau=est_i+1}^{t} y_{i\tau} = -1$. Dans le premier cas, ceci signifie que l'activité n'a pas encore débuté et les inégalités imposent que $b_{it} \leq 0$ et donc $b_{it} = 0$. Dans le second cas, l'équation devient $-r_i^{max} \geq b_{it}$, ce qui est impossible. Notons que ce dernier cas nous assure ainsi qu'une activité ne peut finir avant d'avoir commencé. Cependant, afin de renforcer la relaxation linéaire du modèle, des contraintes spécifiques, empêchant que le début d'une activité ne se produise avant sa fin, peuvent être ajoutées. Ces inégalités sont de la forme :

$$\sum_{\tau=est_i}^{t} x_{i\tau} - \sum_{\tau=est_i+1}^{t} y_{i\tau} \leq 0 \quad \forall t \in \{est_i, \ldots, let_i - 1\} \tag{5.13}$$

La contrainte (5.6) modélise le fait qu'une activité doit recevoir au moins la quantité d'énergie requise par la donnée du problème.

La contrainte (5.7) limite la quantité de ressource utilisée simultanément à la capacité de cette dernière.

Enfin, les contraintes (5.8), (5.9) et (5.10) fixent la valeur des variables à zéro en dehors de leurs fenêtres de temps, i.e. $[est_i, let_i]$ pour $b_{it}$, $[est_i, lst_i]$ pour $x_{it}$ et $[eet_i, let_i]$ pour $y_{it}$.

Cette formulation possède $2nT$ variables binaires, $nT$ variables continues et au plus $3n+T*(5n+1)$ contraintes.

**Fonction de rendement affine** Dans le cas où les fonctions de rendement sont toutes la fonction identité, l'énergie apportée à une activité durant une certaine période est égale à la quantité de ressource consommée par celle-ci durant la même période de temps. Or, lorsque les fonctions de rendement deviennent affines, ce n'est plus le cas. Pour s'assurer qu'une activité $i$ reçoive bien l'énergie requise, nous devons déclarer une nouvelle variable qui permettra de mesurer l'énergie apportée à cette dernière durant la période $t$, $w_{it}$, $\forall (i,t) \in \mathcal{A} \times \mathcal{H}$. Dans le cas précédent, nous avions $w_{it} = b_{it}$.

Nous devons donc vérifier, pour chacune des contraintes du modèle précédent impliquant la variable $b_{it}$, si cette dernière modélisait une quantité de ressource ou d'énergie, i.e. si nous devons la remplacer par $w_{it}$. De plus, une contrainte liant les variables $b_{it}$ et $w_{it}$ devra être rajoutée pour assurer la bonne conversion entre les deux quantités.

La contrainte (5.4) (resp. (5.5)) représentant la quantité minimale (resp. maximale) de ressource qu'une activité doit consommer à chaque instant de son exécution, reste donc inchangée. De même, la contrainte (5.7), modélisant la contrainte sur la capacité de la ressource, n'a pas besoin d'être modifiée.

La contrainte (5.6), qui s'assure qu'une activité reçoive bien la quantité d'énergie requise doit être réécrite en utilisant la variable $w_{it}$. Ce qui nous donne l'inégalité suivante :

$$\sum_{t=est_i}^{let_i} w_{it} \geq W_i \quad \forall i \in \mathcal{A} \tag{5.6'}$$

De plus, nous devons ajouter une contrainte permettant de lier la quantité de ressource consommée par une activité et la quantité d'énergie apportée à celle-ci. Nous ajoutons donc la contrainte suivante au modèle :

$$w_{it} = a_i b_{it} + c_i \left( \sum_{\tau=est_i}^{t} x_{i\tau} - \sum_{\tau=est_i+1}^{t} y_{i\tau} \right) \quad \forall t \in \mathcal{H},\ \forall i \in \mathcal{A} \tag{5.14}$$

Cette contrainte nous permet de modéliser la fonction de rendement $f_i,\ \forall i \in \mathcal{A}$. En effet, $\left(\sum_{\tau=est_i}^{t} x_{i\tau} - \sum_{\tau=est_i+1}^{t} y_{i\tau}\right)$ est égale à 1 si et seulement si l'activité $i$ est en cours à l'instant $t$. Dans ce cas-là, la valeur de l'énergie apportée à $i$ est bien $w_{it} = a_i b_{it} + c_i$. Le second cas se produit quand l'activité $i$ n'est pas en cours à $t$. Dans ce cas, la contrainte (5.5) nous dit que $b_{it} = 0$ et donc $w_{it} = 0$.

Le modèle possède donc $2nT$ variables binaires, $2nT$ variables continues et au plus $3n + T*(6n+1)$ contraintes.

**Fonction de rendement concave affine par morceaux** Lorsque les fonctions de rendements sont des fonctions concaves affines par morceaux, nous utilisons aussi la variable $w_{it}$ pour représenter l'énergie reçue par l'activité $i$ dans la période $t$. La contrainte (5.6') est utilisée pour modéliser le fait que cette activité doive recevoir une quantité d'énergie $W_i$ durant son exécution.

Pour s'assurer de la bonne conversion entre la quantité de ressource consommée par une activité $i$ dans une période $t$ et la quantité d'énergie reçue par cette dernière dans la même période, l'égalité (A.6) est remplacée par l'inégalité suivante :

$$w_{it} \leq a_{ip} b_{it} + c_{ip} \left( \sum_{\tau=est_i}^{t} x_{i\tau} - \sum_{\tau=est_i+1}^{t} y_{i\tau} \right) \quad \forall i \in \mathcal{A},\ \forall p \in \mathcal{P}_i,\ \forall t \in \mathcal{H} \tag{5.15}$$

avec $\mathcal{P}_i$ l'ensemble des intervalles de définition de la fonction $f_i$.

Notons que l'utilisation d'une inégalité peut impliquer que la variable $w_{it}$ ne représente pas exactement la quantité d'énergie apportée à $i$ dans la période $t$. Cependant, la contrainte nous assure que $w_{it} \leq f_i(b_{it})$. De ce fait, à une solution optimale donnée par le modèle correspond toujours une solution optimale du CECSP à dates de début et de fin entières. En effet, supposons que dans une solution

optimale renvoyée par le PLNE, il existe $(i,t)$ tel que $w_{it} < f_i(b_{it})$, alors, deux cas sont possibles :

— $r_i^{min} < f_i^{-1}(w_{it})$ et dans ce cas-là, $b_{it}$ peut prendre la valeur $f_i^{-1}(w_{it})$ et cette solution aura un coût, i.e. une consommation totale de ressource, plus faible que la solution précédente et ceci contredit l'optimalité de la solution.

— $r_i^{min} \geq f_i^{-1}(w_{it})$ et dans ce cas-là, $b_i(t)$ prend la valeur $r_i^{min}$ et $et_i = \min\{t \in \mathbb{N} \mid \int_{\tau=st_i}^{t} f_i(b_i(t))dt \leq W_i\}$. Ici, deux cas sont possibles : le premier correspond au cas où $et_i = \{t \mid y_{it} = 1\}$ et la solution renvoyée par le PLNE est optimale ; dans le second cas $et_i < \{t \mid y_{it} = 1\}$ mais alors, $y_{it-1}$ peut prendre la valeur 1 et $y_{it}$ la valeur 0 et cette solution est de plus faible coût que la solution précédente et ceci contredit l'optimalité de la solution renvoyée par le PLNE.

Le modèle ainsi défini possède alors $2nT$ variables binaires, $2nT$ variables continues et au plus $3n + (5n + nP + 1)T$ contraintes, avec $P = \max_{i \in A} |\mathcal{P}_i|$.

### 5.1.2 Modèles à événements

Dans cette sous-section, nous proposons deux formulations à événements pour le CECSP. Ces deux formulations sont grandement inspirées des formulations start/end et on/off du RCPSP, présentées dans la section 4.2 et issues de [KALM11]. Comme dans le cas du RCPSP, ces modèles se justifient par leur nombre polynomial de variables et de contraintes, ce qui peut s'avérer très pertinent dans la cas de grands horizons de temps. Un argument supplémentaire vient s'ajouter pour le CECSP. En effet, il peut arriver qu'une instance de ce problème ne possède que des solutions à valeurs non entières et ce, même si toutes les données sont entières (voir exemple 1.2.3, page 33).

Comme dans le cas de la formulation à temps discret, ici, les dates de fin des activités ne sont plus totalement définies par leur date de début. De ce fait, dans les formulations à événements du CECSP, nous avons besoin de modéliser deux types d'événements, les débuts et les fins des activités, soit, au plus, $2n$ événements. Ces événements, indexés par l'ensemble $\mathcal{E} = \{1, \ldots, 2n\}$, sont représentés par un ensemble de variables continues, notées $t_e$.

Nous rappelons aussi les notations suivantes : $T = \max_{i \in \mathcal{A}} let_i$ est une borne supérieure sur la date de fin du projet ; $[est_i, lst_i]$, avec $lst_i = let_i - W_i / f_i(r_i^{max})$, est la fenêtre de temps dans laquelle l'activité $i$ peut débuter ; de même, $[eet_i, let_i]$, avec $eet_i = est_i + W_i / f_i(r_i^{max})$, la fenêtre de temps durant laquelle l'activité $i$ peut finir.

Nous allons présenter, en premier lieu, la formulation start/end du CECSP avec fonctions de rendement identité dont nous dériverons les cas de fonctions de rendement affines et affines par morceaux. Ensuite, nous présenterons la formulation on/off du même problème et de ses dérivés.

#### Modèle start/end

Dans le modèle start/end, deux variables binaires $x_{ie}$ et $y_{ie}$, $\forall (i,e) \in \mathcal{A} \times \mathcal{E}$, servent à affecter les dates des différents événements, modélisés par les variables $t_e$, aux débuts et fins des activités. En effet, la variable $x_{ie}$ prend la valeur 1 si et seulement si l'événement $e$ correspond au début de l'activité $i$, i.e. l'activité $i$ commence à la date $t_e$, et est égale à 0 sinon. De même, la variable $y_{ie}$ est égale à 1 si et seulement si l'événement $e$ correspond à la fin de l'activité $i$, et vaut 0 sinon.

**Fonction de rendement identité** Dans le cas où toutes les fonctions de rendement sont la fonction identité, un seul ensemble de variables est nécessaire pour modéliser la consommation de la ressource. En effet, comme la quantité de ressource consommée par une activité $i$ est égale à la quantité d'énergie apportée à cette même activité, il n'est pas utile de définir une variable représentant cette quantité d'énergie.

Un ensemble de variables supplémentaires, $b_{ie}$, $\forall (i,e) \in \mathcal{A} \times \mathcal{E} \setminus \{2n\}$, est donc défini. La variable $b_{ie}$ représente la quantité de ressource consommée par une activité $i$ entre les dates $t_e$ et $t_{e+1}$. Ceci nous permet de modéliser le problème de la façon suivante :

$$\min \sum_{i \in A} \sum_{e \in \mathcal{E} \setminus \{2n\}} b_{ie} \tag{5.16}$$

$$t_e \leq t_{e+1} \qquad \forall e \in \mathcal{E} \setminus \{2n\} \tag{5.17}$$

$$\sum_{e \in \mathcal{E}} x_{ie} = 1 \qquad \forall i \in A \tag{5.18}$$

$$\sum_{e \in \mathcal{E}} y_{ie} = 1 \qquad \forall i \in A \tag{5.19}$$

$$x_{ie} est_i \leq t_e \qquad \forall i \in A,\ \forall e \in \mathcal{E} \tag{5.20}$$

$$t_e \leq x_{ie} lst_i + (1 - x_{ie})\, T \qquad \forall i \in A,\ \forall e \in \mathcal{E} \tag{5.21}$$

$$t_e \geq y_{ie} eet_i \qquad \forall i \in A,\ \forall e \in \mathcal{E} \tag{5.22}$$

$$let_i y_{ie} + (1 - y_{ie})\, T \geq t_e \qquad \forall i \in A,\ \forall e \in \mathcal{E} \tag{5.23}$$

$$\sum_{i \in A} b_{ie} \leq R\,(t_{e+1} - t_e) \qquad \forall e \in \mathcal{E} \setminus \{2n\} \tag{5.24}$$

$$t_f \geq t_e + (x_{ie} + y_{if} - 1)\, W_i / f_i(r_i^{max}) \qquad \forall i \in A,\ \forall e, f \in \mathcal{E}\ ; f \geq e \tag{5.25}$$

$$\sum_{e \in \mathcal{E} \setminus \{2n\}} b_{ie} = W_i \qquad \forall i \in A \tag{5.26}$$

$$b_{ie} \geq r_i^{min}\,(t_{e+1} - t_e) - M \left(1 - \sum_{f=0}^{e} x_{if} + \sum_{f=0}^{e} y_{if}\right) \qquad \forall i \in A,\ \forall e \in \mathcal{E} \setminus \{2n\} \tag{5.27}$$

$$b_{ie} \leq r_i^{max}\,(t_{e+1} - t_e) \qquad \forall i \in A,\ \forall e \in \mathcal{E} \setminus \{2n\} \tag{5.28}$$

$$\left(\sum_{f=0}^{e} x_{if} - \sum_{f=0}^{e} y_{if}\right) M \geq b_{ie} \qquad \forall i \in A,\ \forall e \in \mathcal{E} \setminus \{2n\} \tag{5.29}$$

$$t_e \geq 0 \qquad \forall e \in \mathcal{E} \tag{5.30}$$

$$b_{ie} \geq 0 \qquad \forall i \in A,\ \forall e \in \mathcal{E} \setminus \{2n\} \tag{5.31}$$

$$x_{ie} \in \{0,1\},\ y_{ie} \in \{0,1\} \qquad \forall i \in A,\ \forall e \in \mathcal{E} \tag{5.32}$$

L'objectif, décrit par l'équation (5.16), est de minimiser la consommation totale de la ressource, i.e. la somme des consommations de toutes les tâches. On peut facilement modifier cet objectif afin de minimiser la date de fin du projet en remplaçant l'objectif par :

$$\min t_{|\mathcal{E}|}$$

La contrainte (5.17) ordonne les événements. La contrainte (5.18) (resp. (5.19)) s'assure que chaque activité ne commence (resp. finisse) qu'une et une seule fois. En effet, chaque début (resp. fin) d'activité

ne peut être associé qu'à un et un seul événement.

Les contraintes (5.20) et (5.21) garantissent que la date d'un événement correspondant au début d'une activité soit bien comprise dans sa fenêtre de temps, i.e. dans l'intervalle $[est_i, lst_i]$. En effet, si l'événement $e$ correspond au début de l'activité $i$, i.e. $x_{ie} = 1$, alors la première inégalité devient $est_i \leq t_e$ et la seconde $t_e \leq lst_i$. Les autres configurations donnent : $0 \leq t_e \leq T$ et ceci est trivialement vérifié. De même, les contraintes (5.22) et (5.23) s'assurent que, si un événement $e$ correspond à la fin d'une activité $i$, alors $t_e$ est bien compris entre $eet_i$ et $let_i$.

La contrainte (5.24) s'assure que, dans la période de temps comprise entre $t_e$ et $t_{e+1}$, la quantité de ressource consommée n'excède pas la capacité de cette dernière.

La contrainte (5.25) modélise le fait que si l'activité $i$ commence à l'événement $e$ et se termine à l'événement $f$, alors les dates correspondant à ces deux événements sont au moins séparées par la durée de cette tâche. Or, comme nous ne connaissons pas cette durée, nous utilisons une borne inférieure : $W_i / f_i(r_i^{max})$.

La contrainte (5.26) modélise le fait qu'une activité doit recevoir au moins la quantité d'énergie requise par la donnée du problème.

Les contraintes (5.27) et (5.28) permettent de s'assurer que la consommation instantanée de chaque activité est bien comprise entre $r_i^{min}$ et $r_i^{max}$ durant toute sa durée d'exécution. Pour s'en assurer, il suffit de remarquer que, dans la première inégalité, $\sum_{f=0}^{e} x_{if} - \sum_{f=0}^{e} y_{if} = 1$ si et seulement si l'activité $i$ est en cours entre les événements $e$ et $e+1$. L'inégalité devient donc $b_{ie} \geq r_i^{min}(t_{e+1} - t_e)$. Les autres configurations possibles donnent $b_{ie} \geq r_i^{min}(t_{e+1} - t_e) - M$, avec $M$ une constante suffisamment grande pour dépasser $r_i^{min}(t_{e+1} - t_e)$.

La contrainte (5.29) garantit que si l'activité $i$ n'est pas en cours entre les événements $e$ et $e+1$, alors $b_{ie} = 0$. En effet, si l'activité $i$ n'est pas en cours, i.e. $\sum_{f=0}^{e} x_{if} - \sum_{f=0}^{e} y_{if} = 0$, la contrainte s'écrit $0 \geq b_{ie}$ et donc on a bien $b_{ie} = 0$. Dans le cas où l'activité $i$ est en cours, i.e. $\sum_{f=0}^{e} x_{if} - \sum_{f=0}^{e} y_{if} = 1$, la contrainte s'écrit $M \geq b_{ie}$ et, si $M$ est une constante suffisamment grande pour servir de borne supérieure à $b_{ie}$, alors la contrainte est valide. Notons aussi que ces contraintes empêchent qu'une activité ne puisse commencer avant d'avoir fini. De plus, pour renforcer la formulation, nous pouvons ajouter les contraintes (4.18) modélisant explicitement le fait qu'un événement "début d'activité" se produit forcément avant l'événement "fin d'activité" lui correspondant.

Cette formulation possède $4n^2$ variables binaires, $2n^2 + n$ variables continues et $2n^3 + 13n^2 + 4n - 2$ contraintes.

**Fonction de rendement affine** Pour adapter cette formulation au cas des fonctions de rendement affines, nous avons, dans un premier temps, besoin de définir une variable $w_{ie}$, $\forall i \in \mathcal{A}$ et $\forall e \in \mathcal{E} \setminus \{2n\}$. Cette variable sert à modéliser la quantité d'énergie apportée à une activité $i$ durant l'intervalle de temps $[t_e, t_{e+1}]$. De plus, nous devons identifier les contraintes impliquant la variable $b_{ie}$ pour lesquelles cette variable doit être remplacée par la variable $w_{ie}$. En d'autres termes, nous devons identifier les contraintes traitant de la ressource et les contraintes liées à l'énergie.

La contrainte (5.24), modélisant la contrainte de capacité de la ressource, n'est pas modifiée. De même, les contraintes (5.27) et (5.28), garantissant la contrainte de consommation maximale et mini-

male de la ressource, restent inchangées. Enfin, la contrainte (5.29), s'assurant de la non consommation de ressource des activités en dehors de leur exécution, ne nécessite aucun changement.

La contrainte (5.26) cependant, étant en charge du respect de l'apport requis en énergie de chaque activité, doit être réécrite en utilisant la variable adéquate, $w_{ie}$. Ceci donne l'inégalité suivante :

$$\sum_{e \in \mathcal{E} \setminus \{2n\}} w_{ie} = W_i \quad \forall i \in \mathcal{A} \tag{5.26'}$$

Il nous reste à écrire les contraintes permettant de lier les deux variables, i.e. de s'assurer de la bonne conversion ressource/énergie. Nous ajoutons donc les contraintes suivantes au modèle :

$$w_{ie} \leq a_i b_{ie} + c_i(t_{e+1} - t_e) \qquad \forall i \in A,\ \forall e \in \mathcal{E} \setminus \{2n\} \tag{5.33}$$

$$w_{ie} \leq W_i(\sum_{f=0}^{e} x_{if} - \sum_{f=0}^{e} y_{if}) \qquad \forall i \in A,\ \forall e \in \mathcal{E} \setminus \{2n\} \tag{5.34}$$

$$w_{ie} \geq 0 \qquad \forall i \in A,\ \forall e \in \mathcal{E} \setminus \{2n\} \tag{5.35}$$

La contrainte (5.33) permet de modéliser l'énergie apportée à l'activité $i$ en fonction de la quantité de ressource consommée par cette même activité dans l'intervalle $[t_e, t_{e+1}]$. Notons d'abord l'utilisation d'une inégalité et non d'une égalité. Ceci est dû au fait que, lorsque l'activité n'est pas en cours entre $t_e$ et $t_{e+1}$, i.e. $b_{ie} = 0$, la contrainte devient $w_{ie} \leq c_i(t_{e+1} - t_e)$. Dans le cas d'une égalité, ceci aurait été faux puisqu'on devrait avoir $w_{ie} = 0$. Le rôle de la seconde contrainte est donc de s'assurer que lorsque l'activité n'est pas en cours, on ait bien $w_{ie} = 0$. En effet, lorsque l'activité n'est pas en cours, la contrainte devient $w_{ie} \leq 0$ et dans le cas contraire, l'inégalité s'écrit $w_{ie} \leq W_i$.

La seconde remarque qui peut être faite est que, puisque la contrainte (5.33) utilise une inégalité, $w_{ie}$ peut ne pas être réellement égale à la quantité d'énergie apportée à $i$ entre $t_e$ et $t_{e+1}$. En fait, grâce à l'objectif (5.16), ceci ne peut arriver.

**Théorème 5.1.** *Soit $S$ une solution optimale du modèle décrit par (5.16)–(5.25), (5.26') et (5.27)–(5.35). Alors $S$ vérifie la condition suivante :*

$$\forall i \in \mathcal{A},\ \forall e \in \mathcal{E} \setminus \{2n\},\ w_{ie} = f_i\left(\frac{b_{ie}}{t_{e+1} - t_e}\right)(t_{e+1} - t_e)$$

*Démonstration.* Pour prouver le théorème, commençons par remarquer que la contrainte (5.33) implique $w_{ie} \leq f_i\left(\frac{b_{ie}}{t_{e+1}-t_e}\right)(t_{e+1} - t_e)$. En effet,

$$\begin{aligned} f_i\left(\frac{b_{ie}}{t_{e+1} - t_e}\right)(t_{e+1} - t_e) &= \left(a_i \frac{b_{ie}}{t_{e+1} - t_e} + c_i\right)(t_{e+1} - t_e) \\ &= a_i b_{ie} + c_i(t_{e+1} - t_e) \end{aligned}$$

Il nous reste donc à montrer que $w_{ie} \geq f_i\left(\frac{b_{ie}}{t_{e+1}-t_e}\right)(t_{e+1} - t_e)$. Par l'absurde, supposons que $\exists i \in \mathcal{A},\ \exists e \in \mathcal{E} \setminus \{2n\}$ tel que $w_{ie} < f_i\left(\frac{b_{ie}}{t_{e+1}-t_e}\right)(t_{e+1} - t_e)$. Soit $\mathcal{S}_C$ l'ensemble des solutions du CECSP et soit $c : \mathcal{S}_C \to \mathbb{R}$ la fonction qui associe à une solution $S$ sa consommation de ressource $c(S)$. Nous allons créer une nouvelle solution $S'$ qui vérifie que $c(S) > c(S')$, ce qui invalidera l'optimalité de $S$.

Pour cela, nous considérons la solution du CECSP associée à $S$. Notons $\widehat{S}$ cette solution. $\widehat{S}$ est obtenue à partir de $S$ de la manière suivante, $\forall (i,e) \in \mathcal{A} \times \mathcal{E}$ :

— si $x_{ie} = 1$ alors $st_i$ est fixé à $t_e$,
— si $y_{ie} = 1$ alors $et_i$ est fixé à $t_e$,
— $b_i(t) = \frac{b_{ie}}{t_{e+1}-t_e}, \forall t \in [t_e, t_{e+1}]$.

$S'$ est définie de la façon suivante :

— $st'_i = st_i$,
— $et'_i = \min(t \in \mathcal{H} | \int_{st_i}^{t} f_i(b_i(t)) = W_i)$,
— $b'_i(t) = \begin{cases} b_i(t) & \text{si } st'_i \leq t \leq et'_i \\ 0 & \text{sinon} \end{cases}$

Clairement, $S'$ est une solution pour le CECSP. Nous montrons maintenant que $c(S) = c(\widehat{S}) > c(S')$. Pour cela, nous allons montrer qu'il existe une tâche $i$ telle que $et_i > et'_i$ et donc $\int_{st_i}^{et_i} b_i(t)dt > \int_{st'_i}^{et'_i} b_i(t)dt$. De ce fait, nous aurons bien que $\sum_{i\in\mathcal{A}} \int_{st_i}^{et_i} b_i(t)dt > \sum_{i\in\mathcal{A}} \int_{st'_i}^{et'_i} b_i(t)dt$.

On sait que $\exists (i,e) \in \mathcal{A} \times \mathcal{E}$ tel que :

$$
\begin{aligned}
& w_{ie} < f_i\left(\frac{b_{ie}}{t_{e+1}-t_e}\right)(t_{e+1}-t_e) \\
\Rightarrow & \sum_{e\in\mathcal{E}} w_{ie} < \sum_{e\in\mathcal{E}} f_i\left(\frac{b_{ie}}{t_{e+1}-t_e}\right)(t_{e+1}-t_e) \\
\Rightarrow & W_i = \sum_{e\in\mathcal{E}} w_{ie} < \sum_{e\in\mathcal{E}} f_i(b_i(t))(t_{e+1}-t_e) = \int_{st_i}^{et_i} f_i(b_i(t))dt \\
\Rightarrow & \min\left(t \in \mathcal{H} | \int_{st_i}^{t} f_i(b_i(t))dt\right) = et'_i < et_i
\end{aligned}
$$

Donc nous avons bien $et_i > et'_i$ et ceci achève la démonstration. □

Notons que pour d'autres objectifs cela peut ne pas être valide. Mais, la solution obtenue par le modèle peut toujours être transformée en une solution du CECSP en temps polynomial. Pour cela, il suffit de suivre la transformation proposée dans la preuve du théorème 5.1.

Le modèle ainsi défini possède $4n^2$ variables binaires, $4n^2$ variables continues et $2n^3+17n^2+2n-2$ contraintes.

**Fonction de rendement affine par morceaux** Lorsque les fonctions de rendement sont des fonctions concaves affines par morceaux, nous utilisons aussi la variable $w_{ie}$ pour représenter l'énergie reçue par l'activité $i$ dans la période $[t_e, t_{e+1}]$. La contrainte (5.26') est utilisée pour modéliser le fait que cette activité doive recevoir une quantité d'énergie $W_i$ durant son exécution.

Pour s'assurer de la bonne conversion entre la quantité de ressource consommée par une activité $i$ dans une période $[t_e, t_{e+1}]$ et la quantité d'énergie reçue par cette dernière dans la même période, l'inégalité (5.33) est remplacée par l'inégalité suivante :

$$
w_{ie} \leq a_{ip} b_{ie} + c_{ip}\,(t_{e+1} - t_e) \quad \forall i \in \mathcal{A},\ \forall p \in \mathcal{P}_i,\ \forall e \in \mathcal{E} \setminus \{2n\} \tag{5.36}
$$

Notons que la même argumentation que celle de la preuve du théorème 5.1 permet de nous assurer

que la conversion faite par le modèle entre la quantité d'énergie reçue par l'activité et la quantité de ressource consommée par cette dernière est valide.

Le modèle ainsi défini possède $4n^2$ variables binaires, $4n^2$ variables continues et $2n^3+(15+2P)n^2+(1-P)\,n-2$ contraintes, avec $P = \max_{i\in A} |\mathcal{P}_i|$.

### Modèle on/off

Dans le modèle on/off, une variable binaire $z_{ie}$ se charge d'assigner à chaque événement les dates de début et de fin des activités. La variable $z_{ie}$ prendra la valeur 1 si et seulement si l'activité $i$ est en cours d'exécution dans la période $[t_e, t_{e+1}]$ et sera égale à 0 sinon. Ceci permet de diviser par deux le nombre de variables binaires utilisées par rapport à la formulation start/end.

**Fonction de rendement identité** Dans le cas où les fonctions de rendement sont la fonction identité, nous définissons la variable $b_{ie}$ qui modélise la quantité de ressource consommée par l'activité $i$ entre les dates $t_e$ et $t_{e+1}$. Comme pour le modèle start/end, nous avons seulement besoin de définir la variable $b_{ie},\ \forall i \in \mathcal{A}$ et $\forall e \in \mathcal{E} \setminus \{2n\}$. Ceci conduit à la formulation suivante :

$$\min \sum_{i\in A} \sum_{e\in\mathcal{E}\setminus\{2n\}} b_{ie} \tag{5.37}$$

$$t_e \le t_{e+1} \qquad \forall e \in \mathcal{E} \setminus \{2n\} \tag{5.38}$$

$$\sum_{e\in\mathcal{E}} z_{ie} \ge 1 \qquad \forall i \in \mathcal{A} \tag{5.39}$$

$$est_i z_{ie} \le t_e \le lst_i(z_{ie} - z_{i,e-1}) + (1 - (z_{ie} - z_{i,e-1}))T \qquad \forall e \in \mathcal{E} \setminus \{1\},\ \forall i \in \mathcal{A} \tag{5.40}$$

$$eet_i(z_{i,e-1} - z_{ie}) \le t_e \qquad \forall e \in \mathcal{E} \setminus \{1\},\ \forall i \in \mathcal{A} \tag{5.41}$$

$$t_e \le let_i(z_{i,e-1} - z_{ie}) + (1 - (z_{i,e-1} - z_{ie}))T \qquad \forall e \in \mathcal{E} \setminus \{1\},\ \forall i \in \mathcal{A} \tag{5.42}$$

$$t_f \ge t_e + ((z_{ie} - z_{i,e-1}) - (z_{if} - z_{i,f-1}) - 1)W_i/f_i(r_i^{max}) \qquad \forall e, f \in \mathcal{E},\ f > e \ne 1,\ \forall i \in \mathcal{A} \tag{5.43}$$

$$\sum_{e'=1}^{e} z_{ie'} \le e(1 - (z_{ie} - z_{i,e-1})) \qquad \forall e \in \mathcal{E} \setminus \{1\},\ \forall i \in \mathcal{A} \tag{5.44}$$

$$\sum_{e'=e}^{2n} z_{ie'} \le (2n - e)(1 + (z_{ie} - z_{i,e-1})) \qquad \forall e \in \mathcal{E} \setminus \{1\},\ \forall i \in \mathcal{A} \tag{5.45}$$

$$\sum_{i\in\mathcal{A}} b_{ie} \le R(t_{e+1} - t_e) \qquad \forall e \in \mathcal{E} \setminus \{2n\} \tag{5.46}$$

$$\sum_{e\in\mathcal{E}\setminus\{2n\}} b_{ie} = W_i \qquad \forall i \in \mathcal{A} \tag{5.47}$$

$$b_{ie} \ge r_i^{min}(t_{e+1} - t_e) - M(1 - z_{ie}) \qquad \forall e \in \mathcal{E} \setminus \{2n\},\ \forall i \in \mathcal{A} \tag{5.48}$$

$$b_{ie} \le r_i^{max}(t_{e+1} - t_e) \qquad \forall e \in \mathcal{E} \setminus \{2n\},\ \forall i \in \mathcal{A} \tag{5.49}$$

$$z_{ie}M \ge b_{ie} \qquad \forall e \in \mathcal{E} \setminus \{2n\},\ \forall i \in \mathcal{A} \tag{5.50}$$

$$t_e \ge 0 \qquad \forall e \in \mathcal{E} \tag{5.51}$$

$$b_{ie} \ge 0 \qquad \forall i \in A,\ \forall e \in \mathcal{E} \setminus \{2n\} \tag{5.52}$$

$$z_{ie} \in \{0, 1\} \qquad \forall i \in A,\ \forall e \in \mathcal{E} \tag{5.53}$$

Dans cette formulation, l'objectif est décrit par (5.37). Ici, l'objectif est de minimiser la consommation totale de la ressource durant tout le projet. Si l'on souhaite minimiser la date de fin du projet, l'objectif

s'écrit facilement comme :

$$\min t_{|\mathcal{E}|}$$

La contrainte (5.38) joue le même rôle que dans la formulation start/end, à savoir, ordonner les événements.

La contrainte (5.39) stipule que chaque activité doit être ordonnancée une fois dans toute la durée du projet.

La contrainte (5.40) s'assure que la date d'un événement correspondant au début d'une activité soit bien comprise dans sa fenêtre de temps. En effet, si l'événement $e$ correspond au début de l'activité $i$, i.e. $z_{ie} = 1$ et $z_{i,e-1} = 0$ , l'inégalité devient alors : $est_i \leq t_e \leq lst_i$. Notons que pour le cas $e = 1$, $z_{i,e-1} - z_{ie}$ est remplacé par $z_{ie}$. De même, les contraintes (5.41) et (5.42) garantissent qu'un événement correspondant à la fin d'une activité soit bien compris entre $eet_i$ et $let_i$. Ici, il n'est pas nécessaire de considérer le cas $e = 1$ car cet événement ne peut pas correspondre à la fin d'une activité.

La contrainte (5.43) assure une séparation suffisante entre un événement correspondant au début d'une activité et un événement correspondant à la fin de cette même activité. Ici, comme nous ne connaissons pas la durée d'une activité, nous utilisons une borne inférieure sur cette dernière. Pour le cas où $e = 1$, il suffit de remplacer $z_{i,e-1} - z_{ie}$ par $z_{ie}$.

Les contraintes (5.44) et (5.45), appelées *contraintes de contiguïté*, assure la non-préemption des activités. Une preuve formelle de la validité de ces contraintes est décrite dans [KALM11]. Ici aussi, le cas $e = 1$, est traité en remplaçant $z_{i,e-1} - z_{ie}$ par $z_{ie}$.

La contrainte (5.46) représente la capacité de la ressource tandis que la contrainte (5.47) s'assure que chaque activité reçoit bien la quantité d'énergie requise par la donnée du problème.

Les contraintes (5.48), (5.49) et (5.50) – permettant respectivement de modéliser les contraintes de consommation minimale, de consommation maximale et de consommation nulle en dehors de l'intervalle d'exécution de $i$ – se déduisent directement des contraintes (5.27), (5.28) et (5.29). Dans un premier temps, remarquons que nous avons la relation suivante :

$$z_{ie} = \sum_{f=1}^{e} x_{if} - \sum_{f=1}^{e} y_{if} \quad \forall i \in \mathcal{A},\ \forall e \in \mathcal{E} \setminus \{2n\} \tag{5.54}$$

En effet, $z_{ie}$ vaut 1 si et seulement si l'activité $i$ est en cours entre $t_e$ et $t_{e+1}$, c'est-à-dire si l'activité a commencé à l'événement $e$ ou avant, i.e. $\sum_{f=1}^{e} x_{if} = 1$, et si elle ne finit pas à l'événement $e$ ou avant, i.e. $\sum_{f=1}^{e} y_{if} = 0$. Dans ce cas, nous avons bien $\sum_{f=1}^{e} x_{if} - \sum_{f=1}^{e} y_{if} = 1 = z_{ie}$. Les contraintes (5.48), (5.49) et (5.50) s'obtiennent donc facilement à partir de (5.27), (5.28) et (5.29) en remplaçant $\sum_{f=1}^{e} x_{if} - \sum_{f=1}^{e} y_{if}$ par $z_{ie}$.

Le modèle ainsi défini possède $2n^2 - n$ variables binaires, $2n^2 + n$ variables continues et $2n^3 + 13n^2 - n - 2$ contraintes.

**Fonction de rendement affine** Pour modéliser la conversion de la ressource en énergie, nous définissons une variable $w_{ie}$, $\forall (i, e) \in \mathcal{A} \times \mathcal{E} \setminus \{2n\}$, représentant la quantité d'énergie apportée à l'activité $i$ dans l'intervalle $[t_e, t_{e+1}]$. Puis, nous identifions les contraintes du précédent modèle qui

ont besoin d'être réécrites à l'aide de cette variable. Ici, seule la contrainte (5.47) est concernée. La réécriture donne la contrainte suivante :

$$\sum_{e \in \mathcal{E}} w_{ie} = W_i \quad \forall i \in \mathcal{A} \tag{5.47'}$$

Dans un second temps, nous écrivons les contraintes liant les variables $b_{ie}$ et $w_{ie}$ :

$$w_{ie} \leq a_i b_{ie} + c_i(t_{e+1} - t_e) \qquad \forall i \in A,\ \forall e \in \mathcal{E} \setminus \{2n\} \tag{5.55}$$
$$w_{ie} \leq W_i z_{ie} \qquad \forall i \in A,\ \forall e \in \mathcal{E} \setminus \{2n\} \tag{5.56}$$
$$w_{ie} \geq 0 \qquad \forall i \in A,\ \forall e \in \mathcal{E} \setminus \{2n\} \tag{5.57}$$

Notons que la même argumentation que celle de la preuve du théorème 5.1 permet de nous assurer de la bonne conversion entre énergie et ressource du modèle.

Le modèle ainsi défini possède $2n^2 - n$ variables binaires, $4n^2$ variables continues et $2n^3 + 17n^2 - 3n - 2$ contraintes.

**Fonction de rendement affine par morceaux** Lorsque les fonctions de rendement sont des fonctions concaves affines par morceaux, nous utilisons aussi la variable $w_{ie}$ pour représenter l'énergie reçue par l'activité $i$ dans la période $[t_e, t_{e+1}]$. La contrainte (5.26') est utilisée pour modéliser le fait que cette activité doive recevoir une quantité d'énergie $W_i$ durant son exécution.

Pour s'assurer de la bonne conversion entre la quantité de ressource consommée par une activité $i$ dans une période $[t_e, t_{e+1}]$ et la quantité d'énergie reçue par cette dernière dans la même période, l'inégalité (5.33) est remplacée par l'inégalité suivante :

$$w_{ie} \leq a_{ip} b_{ie} + c_{ip}\,(t_{e+1} - t_e) \quad \forall i \in \mathcal{A},\ \forall p \in \mathcal{P}_i,\ \forall e \in \mathcal{E} \setminus \{2n\} \tag{5.58}$$

Le modèle ainsi défini possède $2n^2 - n$ variables binaires, $4n^2$ variables continues et $2n^3 + (15 + 2P)n^2 + (2 - P)\,n - 2$ contraintes, avec $P = \max_{i \in A} |\mathcal{P}_i|$.

Dans la partie suivante, nous allons montrer comment améliorer ces modèles par le biais d'une étude polyédrale et la proposition d'inégalités valides.

## 5.2 Renforcement des modèles

Dans cette section, nous présentons des améliorations mises en place pour les modèles indexés par le temps et les modèles à événements du RCPSP et du CECSP.

Dans un premier temps, nous nous sommes intéressé aux modèles à temps discret pour lesquels nous proposons des inégalités valides déduites directement du raisonnement énergétique présenté à la sous-section 2.2.3 pour le CuSP et à la section 3.2 pour le CECSP. En effet, ces modèles restent les modèles les plus efficaces pour résoudre le CECSP ou le RCPSP, ceci étant principalement dû à la qualité relative de leur relaxation.

Cependant, les limitations de ces modèles nous ont poussé à nous intéresser plus particulièrement

à l'amélioration des modèles à événements. En effet, le nombre de variables et de contraintes des modèles discrets dépend de la taille de l'horizon de temps du problème. Ce nombre peut donc s'avérer très important si l'horizon de temps est grand. De plus, pour le CECSP , la discrétisation de l'horizon de temps conduit à une réduction de l'espace des solutions et peut donc mener à des solutions sous-optimales (voir exemple 1.2.3, page 33).

Pour améliorer les relaxations linéaires des modèles à événements, nous avons étudié le polyèdre défini par l'ensemble de toutes les affectations possibles des variables binaires pour une seule activité, pour lequel nous exhibons un ensemble minimal d'inégalités permettant de le décrire. De plus, plusieurs ensembles d'inégalités valides sont proposées pour contribuer à l'amélioration des performances des modèles à événements. Les améliorations apportées à ces modèles ont été faites en collaboration avec Tamás Kis [NKAL16].

### 5.2.1 Amélioration des modèles indexés par le temps

Dans cette sous-section, nous montrons comment est utilisé le raisonnement énergétique décrit à la section 3.2 pour exhiber des inégalités valides pour les modèles de programmation linéaire mixte indexés par le temps du RCPSP et du CECSP. Nous commençons par détailler le cas du CECSP avant de montrer comment appliquer le même raisonnement pour le RCPSP.

Pour intégrer le raisonnement énergétique dans les modèles indexés par le temps, nous utilisons l'équation qui se trouve au centre de ce raisonnement (cf. équation (2.2)). Soit $\mathcal{I}$ l'ensemble des intervalles d'intérêt du raisonnement énergétique, alors l'équation s'écrit :

$$\begin{aligned} & SL(t_1,t_2) \geq 0 \quad \forall [t_1,t_2] \in \mathcal{I} \\ \Rightarrow \underline{b}(i,t_1,t_2) + \sum_{\substack{j \in \mathcal{A} \\ j \neq i}} & \underline{b}(j,t_1,t_2) \leq R(t_2-t_1) \quad \forall [t_1,t_2] \in \mathcal{I} \end{aligned}$$

Soit $CR : \mathbb{R} \times \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ qui associe à une quantité d'énergie $w$ et à un intervalle $[t_1,t_2]$, la quantité de ressource minimale à consommer dans $[t_1,t_2]$ pour apporter à une activité une énergie $w$ (voir équation (3.21)). En considérant toutes les expressions possibles pour $\underline{b}(i,t_1,t_2)$ et en utilisant les variables $x_{it}$ et $y_{it}$ pour activer ou non l'équation correspondant à la consommation minimale de ressource dans $[t_1,t_2]$, nous obtenons les inégalités suivantes :

$$\left(1 - \sum_{t \leq t_1} x_{it} - \sum_{t \geq t_2} y_{it}\right) CR\left(W_i,t_1,t_2\right) + \sum_{j \neq i} \underline{b}(j,t_1,t_2) \leq R(t_2-t_1)$$
$$\forall i \in \mathcal{A},\ \forall [t_1,t_2] \in \mathcal{I} \qquad (5.59)$$

L'inégalité (5.59) représente le cas où l'intervalle $[est_i, let_i]$ est complètement inclus dans $[t_1,t_2]$. En effet, les différents cas possibles sont :

- l'activité commence après $t_1$ et finit avant $t_2$. Dans ce cas-là, l'inégalité devient : $CR\left(W_i,t_1,t_2\right) + \sum_{j \neq i} \underline{b}(j,t_1,t_2) \leq R(t_2-t_1)$. Ceci correspond au cas où la contrainte est activée.
- l'activité commence avant $t_1$ et finit avant $t_2$. Ici, l'inégalité devient : $\sum_{j \neq i} \underline{b}(j,t_1,t_2) \leq R(t_2-t_1)$ et ceci est vérifié quel que soit $i$ et $[t_1,t_2]$. Ce cas correspond au cas où la contrainte n'est pas

activée.

— l'activité commence après $t_1$ et finit après $t_2$. Ce cas est similaire au précédent.

— l'activité commence avant $t_1$ et finit après $t_2$. L'inégalité s'écrit alors : $\sum_{j\neq i}\underline{b}(j,t_1,t_2) - CR\left(W_i,t_1,t_2\right) \leq R(t_2-t_1)$. Ce cas est aussi similaire au second.

$$\begin{aligned}&\left(x_{i,est_i}+y_{i,let_i}-1\right)\, CR\left(W_i - f_i(r_i^{max})\left(t_1-est_i+let_i-t_2\right),t_1,t_2\right)\\&+\sum_{j\neq i}\underline{b}(j,t_1,t_2) \leq R\left(t_2-t_1\right) \qquad \forall i\in\mathcal{A},\ \forall[t_1,t_2]\in\mathcal{I}\end{aligned} \tag{5.60}$$

L'inégalité (5.60) correspond au cas où l'activité est centrée, i.e. $est_i \leq t_1 < t_2 \leq let_i$ et $W_i - (t_1 - est_i + let_i - t_2) \geq f_i(r_i^{min})(t_2-t_1)$. Dans ce cas, l'inégalité sera activée si et seulement si l'activité commence à $est_i$ ($x_{i,est_i}=1$) et finit à $let_i$ ($y_{i,let_i}=1$).

$$\begin{aligned}&\left(x_{i,est_i}+\sum_{t=t_1}^{t_2}y_{it}-1\right)\, CR\left(W_i - f_i\left(r_i^{max}\right)\left(t_1-est_i\right),t_1,t_2\right)+\\&\sum_{j\neq i}\underline{b}(j,t_1,t_2) \leq R\left(t_2-t_1\right) \qquad \forall i\in\mathcal{A},\ \forall[t_1,t_2]\in\mathcal{I}\end{aligned} \tag{5.61}$$

L'inégalité (5.61) correspond au cas où l'activité est calée à gauche, i.e. exécutée à $r_i^{max}$ pendant l'intervalle $[est_i,t_1]$. Cette inégalité est activée quand l'activité commence à $est_i$ et finit dans l'intervalle $[t_1,t_2]$.

$$\begin{aligned}&\left(\sum_{t=t_1}^{t_2}x_{it}+y_{i,let_i}-1\right)\, CR\left(W_i - f_i\left(r_i^{max}\right)\left(t_2-let_i\right),t_1,t_2\right)\\&+\sum_{j\neq i}\underline{b}(j,t_1,t_2) \leq R\left(t_2-t_1\right) \qquad \forall i\in\mathcal{A},\ \forall[t_1,t_2]\in\mathcal{I}\end{aligned} \tag{5.62}$$

L'inégalité (5.62) correspond au cas où l'activité est calée à droite, i.e. exécutée à $r_i^{max}$ pendant l'intervalle $[t_2,let_i]$. Cette inégalité est activée quand l'activité commence dans l'intervalle $[t_1,t_2]$ et finit a $let_i$.

$$\begin{aligned}&\left(\sum_{t\leq t_1}x_{it}+\sum_{t\geq t_2}y_{it}-1\right)\, r_i^{min}\left(t_2-t_1\right)+\sum_{j\neq i}\underline{b}(j,t_1,t_2)\\&\leq R\left(t_2-t_1\right) \qquad \forall i\in\mathcal{A},\ \forall[t_1,t_2]\in\mathcal{I}\end{aligned} \tag{5.63}$$

Enfin, l'inégalité (5.63) correspond au cas où l'activité est exécutée à $r_i^{min}$ durant l'intervalle $[t_1,t_2]$. Cette inégalité assure que si l'activité commence à $t_1$ ou avant et finit à $t_2$ ou après alors la quantité de ressource disponible dans $[t_1,t_2]$ est suffisante pour exécuter l'activité à $r_i^{min}$ durant tout cet intervalle.

Le nombre d'inégalités ajoutées aux modèles est donc de l'ordre de $5|\mathcal{R}|n$ avec $|\mathcal{R}|$ le nombre d'intervalles d'intérêt du raisonnement énergétique, i.e. $|\mathcal{R}| \in O(n^2)$.

Notons que ces inégalités peuvent aussi être définies dans le cas du RCPSP. Dans ce cas là, nous devons appliquer le même raisonnement pour chaque ressource. Cependant, il y aura seulement trois cas à considérer : l'activité est calée à gauche, l'activité est calée à droite et l'activité est en cours d'exécution durant l'intervalle $[t_1, t_2]$. Ceci donne les inégalités suivantes :

$$\left(x_{i,est_i} + \sum_{t=t_1}^{t_2} y_{it} - 1\right) \underline{b}_{LS}(i, t_1, t_2) + \sum_{j \neq i} \underline{b}(j, t_1, t_2) \leq R_k \, (t_2 - t_1)$$
$$\forall i \in \mathcal{A}, \ \forall k \in \mathcal{R}, \ \forall [t_1, t_2] \in \mathcal{I} \quad (5.64)$$

$$\left(\sum_{t=t_1}^{t_2} x_{it} + y_{i,let_i} - 1\right) \underline{b}_{RS}(i, t_1, t_2) + \sum_{j \neq i} \underline{b}(j, t_1, t_2) \leq R_k \, (t_2 - t_1)$$
$$\forall i \in \mathcal{A}, \ \forall k \in \mathcal{R}, \ \forall [t_1, t_2] \in \mathcal{I} \quad (5.65)$$

$$\left(\sum_{t \leq t_1} x_{it} + \sum_{t \geq t_2} y_{it} - 1\right) \underline{b}_{CS}(i, t_1, t_2) + \sum_{j \neq i} \underline{b}(j, t_1, t_2) \leq R_k \, (t_2 - t_1)$$
$$\forall i \in \mathcal{A}, \ \forall k \in \mathcal{R}, \ \forall [t_1, t_2] \in \mathcal{I} \quad (5.66)$$

avec $\underline{b}_{LS}(i, t_1, t_2)$, $\underline{b}_{CS}(i, t_1, t_2)$ et $\underline{b}_{RS}(i, t_1, t_2)$ comme définis dans la sous-section 2.2.3 pour le CuSP, en remplaçant $b_i$ par $r_{ik}$.

Le nombre d'inégalités ajoutées aux modèles est ici de l'ordre de $3|\mathcal{R}|n$ avec $|\mathcal{R}|$ le nombre d'intervalles d'intérêt du raisonnement énergétique pour le RCPSP.

De telles inégalités ont été proposées dans le cadre du modèle préemptif sur les ensembles admissibles [BD04]. Ces inégalités sont ajoutées au modèle indexé par le temps du CECSP et du RCPSP et les performances de ces modèles avec ou sans ces inégalités sont évaluées dans le chapitre 6 afin de montrer leur intérêt.

Les modèles à temps discret sont particulièrement efficaces pour le RCPSP et le CECSP. Cependant, ces modèles possèdent des limitations qui sont décrites dans la section suivante.

### 5.2.2 Amélioration des modèles à événements

Dans cette sous-section, nous présentons les améliorations possibles des modèles à événements. Dans un premier temps, nous montrons que le modèle start/end possède de meilleures relaxations que le modèle on/off, puis nous présentons un ensemble d'inégalités que nous pouvons rajouter au modèle on/off pour renforcer sa relaxation. Cet ensemble d'inégalités est ensuite utilisé pour donner une description minimale du polyèdre défini par l'ensemble de toutes les affectations possibles des variables binaires $z_{ie}$ pour une seule activité [NKAL16].

D'autres ensembles d'inégalités, permettant l'ajout de nouvelles contraintes au modèle, sont ensuite présentées, la suppression des coefficients big-$M$ dans les contraintes existantes ou l'amélioration des contraintes existantes. Dans cette sous-section, nous présentons les résultats pour le CECSP car, dans la plupart des cas, le remplacement de $\mathcal{E} \setminus \{2n\} = \{1, \ldots, 2n-1\}$ par $\{1, \ldots, n\}$ suffit à l'adaptation

de la preuve au RCPSP. Cependant, lorsque ce n'est pas le cas ou que le résultat n'a pas été prouvé pour le RCPSP, nous le précisons.

### Comparaison des relaxations linéaires

Pour montrer que le modèle start/end possède de meilleures relaxations linéaires que le modèle on/off, nous commençons par montrer que le modèle start/end possède des relaxations au moins aussi bonnes que celles du modèle on/off. Pour cela, il suffit de montrer que toute solution du modèle start/end, entière ou non, est solution du modèle on/off. Ceci peut être fait en écrivant les variables $z_{ie}$ en fonction des variables $x_{ie}$ et $y_{ie}$ et nous avons la relation suivante, décrite dans la sous-section 5.1.2 :

$$z_{ie} = \sum_{f=1}^{e} x_{if} - \sum_{f=1}^{e} y_{if} \quad \forall i \in \mathcal{A},\ \forall e \in \mathcal{E} \setminus \{2n\} \tag{5.67}$$

Il nous reste maintenant à montrer que les relaxations du modèle on/off ne sont pas aussi bonnes que celles du modèle start/end. Pour cela, nous relâchons les contraintes d'intégrité des deux modèles et nous trouvons une solution pour la relaxation du modèle on/off qui n'en est pas une pour la relaxation du modèle start/end.

Considérons le sous-modèle formé des équations (5.39), (5.44) et (5.45) et le sous-modèle formé des équations (5.18), (5.19), (5.67) et de l'ensemble d'équations suivant : $\sum_{f=1}^{e} y_{if} + \sum_{f=e}^{n} x_{if} \leq 1,\ \forall i \in \mathcal{A}$. Clairement, si nous trouvons une solution pour le premier sous-modèle qui n'en est pas une pour le second, ceci montrera bien que les relaxations du modèle on/off ne sont pas aussi bonnes que celles des modèles start/end. En effet, toute solution d'un des sous-modèles est solution du modèle entier (start/end ou on/off) en considérant l'instance suivante : $\forall i \in \mathcal{A},\ est_i = 0,\ let_i = T,\ r_i^{min} = 0,\ r_i^{max} = 1,\ W_i = 1,\ f_i(b_i(t)) = b_i(t)$ et $B = n$.

Nous nous plaçons dans le cas où le nombre d'activités est de 2. Alors, nous avons $|\mathcal{E}| = 4$. Dans la suite, les tâches sont notées par les indices $a$ et $b$ et les événements par 0, 1, 2 et 3.

Nous constatons tout d'abord que $z_a = (0,6\,;\,0\,;\,0,7\,;\,0)$ est bien une solution du sous-modèle on/off.

Nous devons maintenant montrer que le sous modèle start/end ne possède pas de solution avec $z_a = (0,6\,;\,0\,;\,0,7\,;\,0)$. Pour cela, étudions les contraintes du modèle qui n'impliquent que l'activité $a$ :

$$x_{a0} + x_{a1} + x_{a2} + x_{a3} = 1 \tag{5.68}$$
$$y_{a0} + y_{a1} + y_{a2} + y_{a3} = 1 \tag{5.69}$$
$$y_{a0} + x_{a0} + x_{a1} + x_{a2} + x_{a3} \leq 1 \tag{5.70}$$
$$y_{a0} + y_{a1} + x_{a1} + x_{a2} + x_{a3} \leq 1 \tag{5.71}$$
$$y_{a0} + y_{a1} + y_{a2} + x_{a2} + x_{a3} \leq 1 \tag{5.72}$$
$$y_{a0} + y_{a1} + y_{a2} + y_{a3} + x_{a3} \leq 1 \tag{5.73}$$
$$0.6 = x_{a0} - y_{a0} \tag{5.74}$$

$$0 = x_{a0} - y_{a0} + x_{a1} - y_{a1} \tag{5.75}$$
$$0.7 = x_{a0} - y_{a0} + x_{a1} - y_{a1} + x_{a2} - y_{a2} \tag{5.76}$$
$$0 = x_{a0} - y_{a0} + x_{a1} - y_{a1} + x_{a2} - y_{a2} + x_{a3} - y_{a3} \tag{5.77}$$

Clairement, dans le modèle start/end, nous avons les contraintes suivantes : $x_{a3} = 0$ et $y_{a0} = 0$. Donc, les contraintes (5.74) et (5.77) impliquent que $x_{a0} = 0,6$ et $y_{a3} = 0,7$.

Or, la contrainte (5.75) implique $y_{a1} = 0,6 + x_{a1}$ et la contrainte (5.68) implique $0,6 + x_{a1} + x_{a2} = 1 \Rightarrow x_{a2} = 0,4 - x_{a1}$.

La contrainte (5.71) peut donc s'écrire $0,6 + 2x_{a1} + x_{a2} \leq 1 \Rightarrow 1 + x_{a1} \leq 1 \Rightarrow x_{a1} = 0$.

Ceci implique $x_{a2} = 0,4$ et $y_{a1} = 0,6$, et l'on obtient une contradiction avec la contrainte (5.69) puisque $0 + 0,6 + y_{a2} + 0,7 > 1$.

Donc, nous avons bien montré que le modèle start/end possède en théorie de meilleures relaxations que le modèle on/off. Cependant, si nous ajoutons un ensemble particulier d'inégalités à ce modèle, ceci n'est peut ne pas être le cas. De telles inégalités sont décrites dans le paragraphe suivant.

#### Inégalités de non préemption

Dans ce paragraphe, nous présentons un ensemble d'inégalités que nous utilisons pour donner une description minimale du polyèdre défini par l'ensemble de toutes les affectations possibles des variables binaires $z_{ie}$ pour une seule activité [NKAL16].

L'ensemble d'inégalités, appelées inégalités de non préemption, est défini comme suit. Puisque, dans tout ordonnancement réalisable, chaque activité doit être exécutée sans préemption, $z_{ie}$ doit satisfaire :

$$\sum_{e_u \in \mathcal{F}} (-1)^u z_{i,e_u} \leq 1 \tag{5.78}$$

où $\mathcal{F} = \{e_0, e_1, \ldots, e_{2v}\}$ est un sous-ensemble ordonné de cardinalité impaire de $\mathcal{E}^* = \mathcal{E} \setminus \{2n\}$.

Soit le polyèdre $ZP_i = \{z_i \in [0,1]^{\mathcal{E}^*} \mid z_i \text{ satisfait (5.39) et (5.78)}\}$ et soit $ZQ_i = \text{conv}\{z_i \in \{0,1\}^{\mathcal{E}^*} \mid z_i \text{ satisfait (5.39),(5.44) et (5.45)}\}$. Nous allons montrer le théorème suivant :

**Théorème 5.2.** *$ZP_i = ZQ_i$.*

*Démonstration.* Dans un premier temps, nous rappelons le lemme de Farkas que nous utiliserons plus tard dans la preuve.

**Lemme 5.1** (Lemme de Farkas)**.** *Soit $A$ une matrice de taille $m \times n$ et $b$ un vecteur de $\mathbb{R}^m$. Il existe un vecteur $x \in \mathbb{R}^n$ vérifiant $Ax = b$ si et seulement si pour tout vecteur $y \in \mathbb{R}^m$ tel que $y^T A \leq 0$ on a $y^T b \leq 0$.*

Nous allons exhiber un ensemble d'inégalités linéaires décrivant $ZQ_i$. Pour cela, commençons par

remarquer que les sommets de $ZQ_i$ sont exactement les vecteurs de dimension $|\mathcal{E}^*|$ suivants :

$$z_{i,e}^{uv} = \begin{cases} 1 & \text{si } u \leq e \leq v \\ 0 & \text{sinon} \end{cases} \qquad \forall u,\ v \in \mathcal{E}^*,\ u \leq v$$

Donc, un point $\overline{z}_i \in ZQ_i$ si et seulement si le système linéaire suivant admet une solution réalisable :

$$\sum_{u \leq v} z_{i,e}^{uv} \lambda_{uv} = \overline{z}_{i,e} \qquad \forall e \in \mathcal{E}^* \tag{5.79}$$

$$\sum_{u \leq v} \lambda_{uv} = 1 \tag{5.80}$$

$$\lambda \geq 0 \tag{5.81}$$

Or, d'après le lemme de Farkas, le système (5.79)–(5.81) admet une solution réalisable si et seulement si $\forall \mu \in \mathbb{R}^{|\mathcal{E}^*|}$ tel que :

$$\sum_{e=u}^{v} \mu_e + \mu_0 \leq 0, \quad u \leq v \tag{5.82}$$

$\mu$ satisfait aussi la condition suivante :

$$\sum_{e \in \mathcal{E}^*} \mu_e \overline{z}_{i,e} + \mu_0 \leq 0 \tag{5.83}$$

Donc, puisque les rayons extrêmes du cône caractérisé par (5.82) définissent toutes les inégalités linéaires requises pour décrire $ZQ_i$, il suffit, pour prouver le théorème, de trouver tous ces rayons extrêmes.

Nous allons ensuite montrer qu'il existe une bijection entre les rayons extrêmes du cône (5.82) vérifiant (5.83) et les inégalités de $ZP_i$.

Pour trouver les rayons extrêmes du cône (5.82), il suffit de considérer les trois cas suivants :

**$\mu_0 > 0$.** Quitte à normaliser, nous pouvons suppposer que $\mu_0 = 1$. Dans ce cas-là, pour tout $e \in \mathcal{E}^*$, nous avons $\mu_e \leq -1$. En effet, ceci est déduit de (5.82) en considérant l'équation pour $u = v = e$. Alors, en prenant $\mu_e = -1, \forall e \in \mathcal{E}^*$, (5.83) implique :

$$\sum_{e \in \mathcal{E}^*} -\overline{z}_{i,e} \leq -1$$

D'après le lemme de Farkas, ceci est une inégalité valide pour $ZQ_i$. On peut remarquer que ces inégalités sont équivalentes à (5.39).

**$\mu_0 = 0$.** Alors, nous avons toujours un cône dont les rayons extrêmes sont les vecteurs unité négatifs de $\mathbb{R}^{\mathcal{E}^*}$. Ces rayons extrêmes nous donnent les inégalités $z_{i,e} \geq 0$ qui sont les contraintes de non négativité valides pour $ZQ_i$.

**$\mu_0 < 0$.** Quitte à normaliser, nous pouvons supposer que $\mu_0 = -1$. Nous allons prouver que dans ce cas, il existe une bijection entre les points extrêmes du polyèdre $H$ défini par :

$$\sum_{e=u}^{v} \mu_e \leq 1 \quad u \leq v \tag{5.84}$$

et les inégalités (5.78). Comme (5.84) implique (5.83), ceci montrera bien la bijection entre les rayons extrêmes de (5.82) vérifiant (5.83) et les inégalités (5.78).

Dans un premier temps, nous montrons que le vecteur formé des coefficients de la partie gauche de chaque inégalité de (5.78) est une solution de (5.84) correspondant à un point extrême de $H$.

Soit $\mathcal{F} = \{e_0, e_1, \ldots, e_{2v}\}$ un ensemble d'événements vérifiant $e_i < e_{i+1}$ pour $i = 0, \ldots, 2v-1$. Le vecteur $\overline{\mu}$ formé des coefficients de la partie gauche de chaque inégalité de (5.78) est défini par :

$$\overline{\mu}_e = \begin{cases} (-1)^u & \text{si } e = e_u \in \mathcal{F} \\ 0 & \text{si } e \in \mathcal{E}^* \setminus \mathcal{F} \end{cases}$$

Pour prouver que $\overline{\mu}$ est une solution de (5.84) correspondant à un point extrême de $H$, nous exhibons un sous-système $L$ de (5.84) contenant $|\mathcal{E}^*|$ inégalités linéairement indépendantes telles que chaque inégalité de $L$ soit vérifiée à l'égalité par $\overline{\mu}$.

Le sous-système $L$ est formé des inégalités suivantes :

$$\begin{aligned} &\sum_{e=u}^{e_0} \mu_e \leq 1 && u = 1, \ldots, e_0 - 1 \\ &\sum_{e=e_{2v}}^{u} \mu_e \leq 1 && u = e_{2v}, \ldots, |\mathcal{E}^*| \end{aligned}$$

De plus, $L$ contient l'ensemble d'inégalités suivant, défini pour tout ensemble formé de 3 événements consécutifs $e_{2u}, e_{2u+1}, e_{2u+2} \in \mathcal{F}$ :

$$\begin{aligned} &\sum_{e=e_{2u}}^{e_{2u+2}} \mu_e \leq 1 && \\ &\sum_{e=e_{2u}}^{t} \mu_e \leq 1 && t = e_{2u}, \ldots, e_{2u+2} - 1 \\ &\sum_{e=t}^{e_{2u+2}} \mu_e \leq 1 && t = e_{2u+1} + 1, \ldots, e_{2u+2} - 1 \end{aligned}$$

On peut facilement vérifier que le système ci-dessus est formé de $|\mathcal{E}^*|$ inégalités linéairement indépendantes et que $\overline{\mu}$ vérifie chacune d'entre elle à l'égalité et ceci prouve notre affirmation.

Nous montrons maintenant que toute solution $\overline{\mu}$ correspondant à un point extrême de $H$ équivaut à une inégalité de (5.78). Dans un premier temps, remarquons que la matrice formée par les coefficients de la partie gauche de (5.84) est totalement unimodulaire. En effet, les colonnes de cette matrice peuvent être réordonnées de telle sorte que chaque ligne ne contienne que des 1 consécutifs. Donc, tout sommet de ce polyèdre est un vecteur à valeurs entières. Nous remarquons aussi que $\overline{\mu}_e \leq 1, \ \forall e \in \mathcal{E}$, puisque $\mu_e \leq 1$ est une inégalité de (5.84) $(u = v)$ pour tout $e \in \mathcal{E}^*$.

Soit $u_1$ le premier indice tel que $\overline{\mu}_{u_1} \neq 0$. Nous affirmons que $\overline{\mu}_{u_1} = 1$. Supposons que ce ne soit pas le cas, i.e. $\overline{\mu}_{u_1} \leq -1$ (les coordonnées de $\overline{\mu}$ sont entières). Puisque $\overline{\mu}$ est un point extrême de $H$, il existe un sous-ensemble $L$ formé de $|\mathcal{E}^*|$ inégalités linéairement indépendantes de (5.84) qui sont satisfaites à l'égalité par $\overline{\mu}$.

Remarquons que $L$ doit contenir une inégalité impliquant la variable $\mu_{u_1}$. En effet, si ce n'est pas le cas, cette variable peut être arbitrairement fixée à une valeur négative et toujours satisfaire toutes les inégalités de $L$. Et donc $\overline{\mu}$ ne serait pas un point extrême de $H$ ce qui est une contradiction. Comme $\overline{\mu}_e = 0$ pour $e < u_1$, une telle inégalité doit être de la forme $\sum_{e=u_1}^{v_1} \mu_e \leq 1$. Comme elle doit être satisfaite par $\overline{\mu}$ à l'égalité et que $\overline{\mu}_e \leq 1$, nous avons que $\overline{\mu}$ doit contenir au moins deux coordonnées $q_1$ et $q_2$ tels que $u_1 \leq q_1 \leq q_2 \leq v_1$ avec $\overline{\mu}_{q_1} = \overline{\mu}_{q_2} = 1$ et $\overline{\mu}_e = 0$ pour $q_1 < e < q_2$. Mais, dans ce cas-là, $\overline{\mu}$ violerait l'inégalité $\sum_{e=q_1}^{q_2} \mu_e \leq 1$ et ceci est une contradiction. Donc, la première coordonnée non nulle de $\overline{\mu}$ doit prendre la valeur 1.

La seconde coordonnée non nulle, disons $u_2$, ne peut prendre la valeur 1 par le même raisonnement que précédemment. Donc, cette valeur doit être négative et entière. Mais, dans ce cas-là, nous pouvons suivre la même argumentation que précédemment pour montrer que $u_2$ doit apparaître dans une des inégalités de $\ell \in L$ et aboutir à la même contradiction que précédemment. Donc, la seconde coordonnée non nulle de $\overline{\mu}$ doit être égale à $-1$. De plus, $\ell$ doit contenir une inégalité impliquant une variable $\mu_{u_3}$ de valeur 1 dans $\overline{\mu}$. En effet, dans le cas contraire, $\overline{\mu}$ ne peut satisfaire $\ell$ à l'égalité. Si nous poursuivons cette argumentation tant que $\mu$ possède des coefficients non nuls après $u_3$, nous reconnaissons la suite de coefficients $1/-1$ présente dans (5.78). □

Nous avons donc défini un ensemble d'inégalités permettant une description complète du polyèdre formé par les vecteurs $z_i$ solution du modèle on/off. Cependant, nous ne pouvons ajouter directement ces inégalités au modèle car leur nombre est exponentiel. Dans le chapitre 6, nous présenterons un algorithme de séparation polynomial qui nous permettra de définir un algorithme de branch-and-cut pour le CECSP et le RCPSP (rappelons que les résultats ci-dessus sont valides dans le cas du RCPSP en posant $\mathcal{E}^* = \{1, \ldots, n\}$).

Le paragraphe suivant présente d'autres inégalités valides pour le CECSP et le RCPSP.

### Autres inégalités valides

Dans ce paragraphe, nous décrivons plusieurs ensembles d'inégalités valides pour le RCPSP et le CECSP. Dans la suite, nous considérons qu'un événement correspond à une et une seule date de début/fin, i.e. $\forall e \in \mathcal{E} \setminus \{1\}$, il existe une et une seule activité $i$ vérifiant $(z_{i,e-1} - z_{ie} = 1) \vee (z_{ie} - z_{i,e-1} = 1)$. Cela est toujours possible puisque $|\mathcal{E}| = 2n$. De plus, l'ajout de cette supposition ne change pas la véracité de ce qui précède.

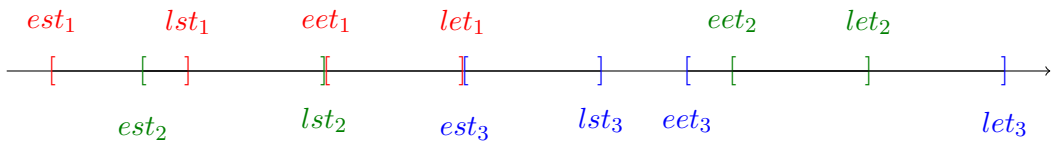
**Séparation maximale entre deux événements** Les inégalités définies dans ce paragraphe sont des inégalités bornant supérieurement la valeur de $t_{e+1} - t_e, \forall e \in \mathcal{E} \setminus \{2n\}$. Pour définir ces inégalités, nous étudions les fenêtres de temps de chaque date de début et de fin d'une activité, i.e. $[est_i, lst_i]$ et $[eet_i, let_i]$, $\forall i \in \mathcal{A}$. L'idée principale repose sur le fait que, dans chacun de ces intervalles, un événement doit forcément avoir lieu. De ce fait, nous savons qu'il y a au moins deux événements consécutifs dans l'union de deux fenêtres de temps consécutives.

Soit $\mathcal{D}$ l'ensemble de toutes les fenêtres de temps, i.e. $\mathcal{D} = \{[est_i, lst_i], [eet_i, let_i],\ \forall i \in \mathcal{A}\}$. Nous commençons par trier les intervalles de $\mathcal{D}$ suivant la règle suivante : $[a, b] \leq [c, d] \Leftrightarrow a < c \vee (a = c \wedge b \leq d)$

Alors, soit $\underline{\mathcal{D}_e}$ (resp. $\overline{\mathcal{D}_e}$) la borne inférieure (resp. supérieure) de l'intervalle $\mathcal{D}_e$, nous avons la propriété suivante :

$$t_{e+1} - t_e \leq \max(\overline{\mathcal{D}_e}, \overline{\mathcal{D}_{e+1}}) - \min(\underline{\mathcal{D}_e}, \underline{\mathcal{D}_{e+1}}) \qquad \forall e \in \mathcal{E} \setminus \{2n\} \tag{5.85}$$

**Exemple 5.2.1.** *Considérons l'ensemble d'intervalles suivant :*




*Après avoir trié les intervalles, nous obtenons :* $[est_1, lst_1] \leq [est_2, lst_2] \leq [eet_1, let_1] \leq [est_3, lst_3] \leq [eet_3, let_3] \leq [eet_2, let_2]$.

*Alors, nous avons l'ensemble de contraintes suivantes :*

- $t_2 - t_1 \leq lst_2 - est_1$
- $t_3 - t_2 \leq let_1 - est_2$
- $t_4 - t_3 \leq lst_3 - eet_1$
- $t_5 - t_4 \leq let_3 - est_3$
- $t_6 - t_5 \leq let_3 - eet_3$

Notons aussi que l'ensemble d'inégalités $\underline{\mathcal{D}_e} \leq t_e \leq \overline{\mathcal{D}_e}$ peut ne pas être valide. En effet, ici $t_6$ peut correspondre à la fin de l'activité 3 et $t_5$ à la fin de l'activité 2, alors que $\mathcal{D}_5 = [eet_3, let_3] \leq [eet_2, let_2] = \mathcal{D}_6$. On aurait alors $\underline{\mathcal{D}_6} \leq t_5 \leq \overline{\mathcal{D}_6}$.

Ces contraintes peuvent être ajoutées au modèle on/off ou utilisées comme borne supérieure sur la valeur de $r_i^{min}(t_{e+} - t_e)$ dans (5.48). L'inégalité se réécrit donc comme :

$$b_{ie} \geq r_i^{min}(t_{e+} - t_e) - r_i^{min}\left(\max(\overline{\mathcal{D}_e}, \overline{\mathcal{D}_{e+1}}) - \min(\underline{\mathcal{D}_e}, \underline{\mathcal{D}_{e+1}})\right)(1 - z_{ie}) \qquad \forall (i, e) \in \mathcal{A} \times \mathcal{E}$$

Ces inégalités peuvent être généralisées à tout sous-ensemble de $k$ intervalles ordonnés $\{\mathcal{D}_{e_1}, \ldots, \mathcal{D}_{e_k}\}$ avec $t_{e_k} - t_{e_1} \leq \max(\overline{\mathcal{D}_{e_1}}, \overline{\mathcal{D}_{e_k}}) - \min(\underline{\mathcal{D}_{e_k}}, \underline{\mathcal{D}_{e_1}})$.

**Date maximale d'un événement** Une idée similaire à celle décrite dans le paragraphe précédent peut être utilisée pour ordonner les événements et calculer des bornes supérieures sur leur date.

Pour faire cela, nous commençons par trier les bornes supérieures des fenêtres de temps de chaque activité, i.e. $lst_i$ et $est_i$, $\forall i \in \mathcal{A}$, par ordre croissant. Alors, puisqu'un événement doit avoir lieu dans chaque fenêtre de temps, i.e. avant chaque borne supérieure de chaque fenêtre, nous pouvons déduire une borne supérieure sur la date de chaque événement.

En effet, soit $\mathcal{UP}$ l'ensemble formé de toutes les bornes supérieures de toutes les fenêtres de temps. Alors, nous avons la propriété suivante :

$$t_e \leq \mathcal{UP}_e \qquad \forall e \in \mathcal{E} \tag{5.86}$$

**Exemple 5.2.2.** *Considérons les intervalles définis dans l'exemple 5.2.1. Alors, nous pouvons déduire l'ensemble de contraintes suivantes :*

- $t_1 \leq lst_1$
- $t_2 \leq lst_2$
- $t_3 \leq let_1$
- $t_4 \leq lst_3$
- $t_5 \leq let_2$
- $t_6 \leq let_3$

Comme précédemment, nous pouvons utiliser ces inégalités comme contraintes additionnelles des modèles à événements ou les utiliser à la place de $T$ dans les contraintes (5.40) et(5.42). Les contraintes s'écrivent alors :

$$\begin{aligned} & est_i z_{ie} \leq t_e \leq lst_i(z_{ie} - z_{i,e-1}) + (1 - (z_{ie} - z_{i,e-1}))\mathcal{UP}_e && \forall e \in \mathcal{E} \setminus \{1\},\ \forall i \in \mathcal{A} \\ & t_e \leq let_i(z_{i,e-1} - z_{ie}) + (1 - (z_{i,e-1} - z_{ie}))\mathcal{UP}_e && \forall e \in \mathcal{E} \setminus \{1\},\ \forall i \in \mathcal{A} \end{aligned}$$

Pour le RCPSP, $t_n$ correspond à borne supérieure sur la durée totale du projet, i.e. sur $T$.

**Inégalités valides dérivées du problème de sac-à-dos** Le rendement minimal de chaque activité pouvant être positif, nous pouvons considérer les contraintes de type sac-à-dos suivantes pour tout $e \in \mathcal{E} \setminus \{2n\}$ et les transformer facilement en inégalités valides :

$$\sum_{i \in \mathcal{A}^+} r_i^{min} z_{ie} \leq B \qquad \forall e \in \mathcal{E} \tag{5.87}$$

où $\mathcal{A}^+$ est le sous-ensemble d'activités avec $r_i^{min} > 0$.

**Inégalités de cliques** Les inégalités de clique permettent de modéliser le fait que plusieurs variables binaires $z_{ie}$ ne peuvent prendre la valeur 1 simultanément. Ces inégalités, déjà établies dans le cas du RCPSP [CAVT87a], correspondent aux sous-ensembles disjonctifs d'activités. Elles sont facilement adaptables au cas du CECSP et sont définies de la manière suivante. Soit $C$ un ensemble minimal d'activités ne pouvant s'exécuter en parallèle, i.e. telles que $\sum_{i \in C} r_i^{min} > B$, alors l'ensemble d'inégalités suivantes est valide pour le CECSP :

$$\sum_{i \in C} z_{ie} \leq |C| - 1 \qquad \forall C,\ \forall e \in \mathcal{E} \tag{5.88}$$

Différentes techniques permettant l'intégration des inégalités ci-dessus seront présentées et comparées dans le chapitre 6.

### Conclusion

Dans ce chapitre, nous avons présenté des modèles de programmation linéaire en nombres entiers pour le CECSP. Pour ce problème, trois modèles sont présentés, un modèle utilisant une discrétisation de l'horizon de temps et deux modèles basés sur une représentation des événements pertinents du problème.

Enfin, des améliorations de ces modèles sont proposées dans la dernière partie du chapitre. Ces améliorations sont basées sur le raisonnement énergétique, la mise en place d'inégalités valides et des études polyédrales. De plus, les avantages et inconvénients de chacun des modèles sont décrits ce qui permet de justifier l'intérêt de ces améliorations.

Des résultats numériques évaluant les performances de ces formulations ainsi que l'intérêt de chaque amélioration sur diverses instances du CECSP et du RCPSP feront l'objet d'un paragraphe dans le chapitre portant sur les expérimentations (cf. Chapitre 6). De plus, un algorithme permettant de séparer les inégalités de non préemption en temps polynomial sera également détaillé dans ce chapitre.

# Conclusion

Cette partie consacrée à la programmation linéaire mixte et en nombres entiers a commencé par décrire les concepts fondamentaux de cette théorie. Puis, dans un second temps, nous avons présenté plusieurs modèles pour un problème d'ordonnancement à contraintes de ressource : le RCPSP. Pour ce problème, plusieurs types de modèles ont été présentés et, pour chacun d'eux, nous avons discuté leurs avantages et inconvénients. Les modèles indexés par le temps sont plus performants sur des instances ayant de petits horizons de temps tandis que les modèles à événements s'avèrent plus efficaces sur des instances disposant de plus grands horizons de temps.

Ces deux types de modèles ont ensuite été adaptés pour être appliqués au CECSP. Pour ce problème, en plus des avantages décrits ci-dessus, les modèles à événements disposent d'un avantage supplémentaire. En effet, pour le CECSP, il est possible qu'une instance ne possède que des solutions rationnelles et ce, même si cette dernière est seulement pourvue de données entières. Les modèles indexés par le temps ne permettent d'obtenir que des solutions à date de début et fin d'activités entières. A l'inverse, les modèles à événements permettant d'obtenir des solutions non entières, sont donc plus adaptés au cas du CECSP. De ce fait, une grande partie de nos travaux a porté sur le renforcement de ces modèles par le biais de l'ajout de coupes et d'inégalités valides. De plus, nous avons montré qu'un de ces ensembles d'inégalités permettait de décrire exactement l'enveloppe convexe des vecteurs binaires solutions d'une partie des contraintes du modèle.

Des inégalités valides pour le modèle indexé par le temps ont aussi été décrites. Ces inégalités sont directement déduites d'un algorithme présenté dans la partie dédiée à la programmation par contrainte de ce manuscrit : le raisonnement énergétique. Dans la continuité de ces travaux, d'autres raisonnements issus de la programmation par contraintes pourraient être envisagés afin de déduire des ensembles d'inégalités pouvant être ajoutées aux modèles (indexé par le temps ou à événements). De plus, les modèles présentés souffrent d'un grand nombre de symétries. Ces dernières pourraient être évitées par l'ajout de nouvelles contraintes. La définition de telles contraintes ainsi qu'une étude du polyèdre associé fait aussi partie des poursuites de recherche sur ce sujet. La dernière perspective dans ce domaine serait l'établissement de nouveaux modèles indexés par le temps, il en existe beaucoup d'autres pour le RCPSP, ou la mise en place de nouvelles formulations étendues.

# Quatrième partie

# Implémentations et Expérimentations


**6 Implémentations et Expérimentations** **143**

6.1 Génération des instances et pré-traitement . . . . . 143

6.1.1 Instances du CECSP . . . . . 143

6.1.2 Instances et pré-calcul des fenêtres de temps pour le RCPSP . . . . . 145

6.2 Performances de la Programmation Linéaire Mixte . . . . . 146

6.2.1 Modèle indexé par le temps . . . . . 146

6.2.2 Modèles à événements . . . . . 147

6.2.3 Comparaison des différentes approches . . . . . 152

6.3 Performances de la Programmation Par Contraintes . . . . . 153

6.3.1 Cadre des expérimentations . . . . . 153

6.3.2 Raisonnement énergétique . . . . . 154

6.3.3 Raisonnements basés sur le Time-Table . . . . . 156

6.3.4 Comparaison des différentes approches . . . . . 157

*La dernière partie de cette thèse est consacrée à la présentation des résultats des expérimentations que nous avons conduites pour valider les méthodes présentées dans les chapitres précédents. Nous commençons, dans un premier temps, par présenter les instances sur lesquelles ces expérimentations ont été menées ainsi que les algorithmes de prétraitement utilisés.*
*La suite de cette partie est ensuite consacrée à la présentation des résultats des méthodes décrites dans le chapitre 5 et issues de la programmation linéaire mixte. Le modèle indexé par le temps pour le CECSP est comparé à ce même modèle auquel on a ajouté les inégalités déduites du raisonnement énergétique et présentées dans la sous-section 5.2.1. Pour les modèles à événements, nous commençons par détailler l'algorithme utilisé pour séparer les inégalités de non préemption (voir sous-section 5.2.2). Les modèles sont ensuite comparés entre eux et l'ajout des inégalités et coupes décrites dans la sous-section 5.2.2 au modèle On/Off dans le cas du CECSP et du RCPSP est discuté. La dernière partie de la section traitant de la programmation linéaire est consacrée à la comparaison des trois modèles.*
*La dernière section concerne les résultats des méthodes issues de la programmation par contraintes présentées dans le chapitre 3. Dans un premier temps, nous décrivons le cadre dans lequel ces expérimentations ont été conduites : les algorithmes de propagation détaillés dans ce manuscrit sont inclus dans une méthode arborescente hybride. Une fois le cadre des expérimentations posé, nous comparons les trois méthodes permettant de calculer les intervalles d'intérêt du raisonnement énergétique. Enfin, la suite de cette partie détaille les résultats de cette méthode arborescente avec les différents algorithmes de propagation.*
*Dû à la difficulté du CECSP, les expérimentations présentées dans cette partie ont majoritairement été conduites sur des instances dans lesquelles les activités ont des fonctions de rendement affines. Quelques résultats sur des instances de petite taille avec des fonctions de rendement concaves et affines par morceaux sont présentés à la fin de cette partie.*

# Chapitre 6

# Implémentations et Expérimentations

Dans ce chapitre, consacré à la présentation des résultats expérimentaux, nous allons, dans un premier temps, décrire les jeux d'instances utilisés ainsi que certaines méthodes de prétraitement qui sont appliquées à ces instances avant leur résolution.

La section suivante sera quant à elle dédiée aux résultats de la programmation linéaire mixte et en nombres entiers. Nous présenterons les performances des différents modèles présentés dans le chapitre 5 ainsi que l'impact des améliorations proposées dans la section 5.2 de ce même chapitre.

Le dernier paragraphe sera consacré à la présentation des performances de la programmation par contraintes où les performances des algorithmes de filtrage présentés dans le chapitre 3 seront comparées.

Tous les tests ont été effectués avec un processeur Intel Core i7-4770 de 3.40GHz, 8 GB de RAM, tournant sous le système d'exploitation Ubuntu 12.04 à 64-bits. Toutes les méthodes présentées ont été codées en C++. Les programmes linéaires mixtes sont résolus avec le solveur commercial ILOG-Cplex version 12.6.

## 6.1 Génération des instances et pré-traitement

Cette section présente, dans un premier temps, la méthode utilisée pour générer des ensembles d'instances de test pour le CECSP. Nous détaillons ensuite les jeux d'instances du RCPSP sur lesquelles nous avons évalué l'impact des améliorations proposées pour les modèles à événements. Nous présenterons aussi une méthode de calcul des fenêtres de temps pour le RCPSP, appliquée sur les instances comme pré-traitement à leur résolution.

### 6.1.1 Instances du CECSP

Les instances utilisées dans le cadre du CECSP sont regroupées en quatre familles selon les caractéristiques de leurs fonctions de rendement et de l'énergie que les activités requièrent. Dans un premier temps, les instances sont générées selon un modèle commun, puis plusieurs types de modifications spécifiques sont appliquées sur celles-ci afin de les séparer en différentes familles.

Tout d'abord, deux ensembles de cinq instances chacun comportant respectivement 10 et 60 activités ainsi que trois ensembles de dix instances chacun comportant respectivement 20, 25 et 30 activités sont générés. Pour toutes ces instances, la disponibilité de la ressource est fixée à $R = 10$ et les autres

paramètres du problème sont générés de façon aléatoire selon une loi uniforme et dans les intervalles suivants :

— $W_i \in [1, \frac{5}{4} * R]$,
— $r_i^{min} \in [0, \frac{1}{4} * W_i]$,
— $r_i^{max} \in [r_i^{min}, 2 * r_i^{min}]$,
— $est_i \in [0, \frac{1}{2} * n]$ et
— $let_i \in [eet_i, eet_i + n]$ avec $eet_i = est_i + \frac{W_i}{f_i(r_i^{max})}$.

Une première famille d'instances, appelée Famille 4, est construite en utilisant comme fonction de rendement la fonction identité pour toutes les activités, i.e. $\forall i \in \mathcal{A},\ f_i(b_i(t)) = b_i(t)$. Ensuite, des fonctions de rendement affines sont générées aléatoirement pour chaque activité. Pour cela, nous générons, pour chaque activité $i \in \mathcal{A}$, les paramètres $a_i$ et $c_i$ de la fonctions $f_i$ suivant une loi uniforme et telle que :

— $a_i \in [1, 10]$ et
— $c_i \in [1, 10]$.

La valeur de $W_i$ est ensuite modifiée à l'aide de ces fonctions $f_i$. Les trois autres familles d'instances sont classées suivant cette modification. Pour la première famille, appelée Famille 1, la nouvelle valeur de $W_i$ est générée aléatoirement, selon une loi uniforme, dans l'intervalle $[0, f_i(W_i)]$ et pour la Famille 2 dans l'intervalle $[f_i(W_i)/2, f_i(W_i)]$. Enfin, pour la Famille 3, $W_i$ est transformé en $f_i(W_i)$.

Pour générer des instances où les activités ont des fonctions de rendement concaves et affines par morceaux, nous utilisons les instances de la Famille 4. Dans un premier temps, nous générons aléatoirement $P_i,\ \forall i \in \mathcal{A}$, le nombre de segments de définition de la fonction $f_i$, dans l'intervalle $[1, r_i^{max} - r_i^{min}]$. L'intervalle $[r_i^{min}, r_i^{max}]$ est ensuite divisé en $P_i$ morceaux. Pour chacun de ces morceaux $p \in \{1, \ldots, P_i\}$, nous générons un coefficient $a_{ip}$ aléatoire tel que $a_{i1} < 10$ et $1 < a_{ip} < a_{ip-1}$ et $c_{ip}$ est calculé de manière à préserver la continuité de la fonction. Enfin, $W_i$ est généré dans l'intervalle $[f_i(r_i^{min})(let_i - est_i), 0, 8 * f_i(r_i^{max})(let_i - est_i)]$. Ce processus est répété deux fois afin d'obtenir deux ensembles de 10 instances chacun, comportant respectivement 10 et 60 activités et trois ensembles de 20 instances chacun comportant 20, 25 et 30 activités. Cette famille d'instance est appelée Famille $LP$.

Afin de comparer les approximations des fonctions de rendement, nous avons aussi généré les instances correspondantes mais avec des fonctions de rendement affines en approximant par le bas les fonctions de rendement affine par morceaux par des fonctions affines. Cette famille d'instances est appelée Famille $L$.

Nous avons donc défini six familles d'instances pour le CECSP. Ces instances serviront à calculer les performances relatives des méthodes présentées dans les chapitres précédents de ce manuscrit.

Dans la prochaine sous-section, nous présentons les instances utilisées pour évaluer les performances des améliorations des modèles du RCPSP ainsi qu'une méthode de calcul des fenêtres de temps.

### 6.1.2 Instances et pré-calcul des fenêtres de temps pour le RCPSP

#### Génération des instances

Les instances utilisées sont les instances définies par Koné [Kon09] pour évaluer les performances des modèles à événements par rapport aux modèles indexés par le temps. Ces instances sont générées à partir des instances à 30 activités de la PSPLIB [KS96]. Ces instances, au nombre de 480, sont des instances à 4 ressources renouvelables et avec des activités ayant une durée aléatoire comprise entre 1 et 10 unités de temps. Ces durées étant relativement courtes, les modèles indexés par le temps sont particulièrement efficaces sur ces dernières (voir sous-section 4.2.2). L'auteur de [Kon09] a donc modifié ces instances afin d'allonger la durée de certaines activités. En effet, dans certaines applications pratiques (notamment dans le domaine pharmaceutique ou de la pétrochimie), il arrive que les activités aient des durées disparates, certaines étant très courtes et d'autres relativement longues.

Les instances de Koné sont créées selon le principe suivant :

1. les 15 premières activités non-fictives de l'instance sont sélectionnées (les autres activités ainsi que les contraintes de précédence qui leur sont adjacentes sont laissées de côté).
2. parmi les activités sélectionnées, celles sans prédécesseurs sont connectées à l'activité 0 et celles sans successeurs à l'activité 16.
3. 7 activités parmi les 15 sélectionnées sont ensuite choisies de façon aléatoire et leur durée est multipliée par un coefficient $25 + b$, où $b$ est un nombre aléatoire généré entre 0 et 1.

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

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

#### Calcul des fenêtres de temps

Afin d'améliorer les performances des modèles pour le RCPSP, nous calculons par prétraitement, pour chaque activité, les fenêtres de temps $[est_i, lst_i]$ et $[eet_i, let_i]$ dans lesquelles elle peut respectivement commencer et finir. Pour calculer ces fenêtres de temps, rappelons que, si chaque arc $(i, j)$ du graphe de précédences $G$ est pondéré par $p_i$, la date de début au plus tôt de $i$, $est_i$ peut prendre la valeur du plus long chemin entre l'activité 0 et l'activité $i$ et la date de début au plus tard $lst_i$ est égale à $est_{n+1}$, la date de début au plus tôt de l'activité terminale $n+1$, moins la valeur du plus long chemin entre $i$ et $n+1$.

Notons que la date de fin au plus tard de l'activité fictive $n+1$ correspondant à une borne supérieure sur la date de fin du projet, elle peut être calculée au moyen d'une heuristique. L'heuristique utilisée pour calculer cette valeur ici est la méthode d'ordonnancement parallèle avec comme règle de priorité la plus petite date de fin au plus tard des activités [Kol96].

En plus des calculs décrits ci-dessus, nous utilisons d'autres techniques de déductions basées sur la propagation de contraintes. En particulier, nous utilisons des techniques décrites dans la thèse de Demassey [Dem03]. Parmi celles utilisées, nous trouvons les techniques de sélection immédiate, Edge-Finding sur les cliques de disjonction et de triplets symétriques. Ces techniques n'étant pas détaillées ici, à l'exception du Edge-Finding à la sous-section 2.2.3, nous renvoyons le lecteur au chapitre 2

de [Dem03] pour une description de ces techniques.

## 6.2 Performances de la Programmation Linéaire Mixte

Dans ce paragraphe, nous détaillons les résultats expérimentaux obtenus par les modèles présentés dans le chapitre 5 et l'influence des techniques de renforcement présentées dans ce même chapitre. Nous commençons, dans un premier temps, par présenter les résultats obtenus pour le modèle indexé par le temps puis nous détaillerons les résultats des modèles à événements.

### 6.2.1 Modèle indexé par le temps

Dans cette sous-section, nous décrivons les performances du modèle indexé par le temps pour le CECSP. Les expérimentations ont été conduites sur les quatre familles d'instances présentées dans la sous-section 6.1.1 et avec une limite de temps de 100 secondes. Le modèle, avec pour objectif la minimisation de la consommation totale de ressource, est dans un premier temps utilisé pour résoudre les instances. Dans un second temps, nous ajoutons les inégalités issues du raisonnement énergétique décrites dans la sous-section 5.2.1. Ces inégalités sont seulement ajoutées au nœud racine de l'arbre de recherche. En effet, dû à leur grand nombre ($5 * |\mathcal{I}| * n$, avec $\mathcal{I}$ l'ensemble des intervalles d'intérêt du raisonnement énergétique) , il est difficile, sans algorithme de séparation, de les inclure dynamiquement pendant la recherche. Le tableau 6.1 présente ces résultats.

Dans ce tableau, nous avons comparé la qualité et le temps d'obtention des premières solutions pour chaque famille d'instances et pour les cas avec et sans coupes énergétiques. La qualité de la solution est mesurée de la manière suivante :

$$\text{gap} = 100 * \frac{|\text{Objectif final} - \text{Objectif } 1^{ère} \text{ Sol}|}{\text{Objectif final}}$$

Nous avons aussi comparé le temps nécessaire à l'obtention de la solution optimale (100 secondes si la solution trouvée n'est pas optimale), la qualité de la solution, le nombre d'instances résolues ($\%solv.$) et le nombre d'entre elles résolues à l'optimum ($\%opt.$).

Les résultats présentés dans le tableau 6.1 montrent que, dans la majorité des cas, le modèle avec les coupes énergétiques obtient une première solution de meilleure qualité que celle obtenue sans ces coupes. De plus, le temps de calcul de cette première solution, bien que légèrement plus élevé dans le cas des coupes énergétiques, est du même ordre de grandeur dans les deux cas.

Cependant, sauf dans le cas des instances à 30 activités de la Famille 4 où l'ajout des coupes permet un gain en termes de performances, les coupes énergétiques ne produisent pas de gain significatif, ni en termes de qualité de solution, ni en termes de temps de résolution.

Malgré tout, les performances relatives des coupes énergétiques pour le calcul des premières solutions pousse à poursuivre cette investigation. La mise en place en place d'un algorithme de séparation permettant de trouver les meilleures coupes à ajouter à chaque nœud de l'arbre de recherche devient donc une direction de recherche encourageante dans un futur proche.

| #act. | $1^{ère}$ sol. | | sol. finale | | | | $1^{ère}$ sol. | | sol. finale | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | tps(s) | gap | tps | gap | %opt. | %solv. | tps | gap | tps | gap | %opt. | %solv. |
| Famille 1 | | | | | | | Famille 2 | | | | | |
| | | | | | sans coupes énergétiques | | | | | | sans coupes énergétiques | |
| 10 | 0,058 | 38 | 45 | 2,9 | 60 | 100 | 0,075 | 24 | 100 | 5,6 | 0 | 100 |
| 20 | 0,25 | 17 | 100 | 8,4 | 0 | 100 | 0,37 | 8,8 | 100 | 6,5 | 0 | 100 |
| 25 | 0,88 | 21 | 100 | 9,4 | 0 | 100 | 2 | 12 | 100 | 5,8 | 0 | 100 |
| 30 | 1,3 | 35 | 100 | 10 | 0 | 100 | 3,1 | 14 | 100 | 6,3 | 0 | 100 |
| | | | | | avec coupes énergétiques à la racine | | | | | | avec coupes énergétiques à la racine | |
| 10 | 0,064 | 3,4 | 60 | 2,6 | 40 | 100 | 0,074 | 9 | 100 | 4,1 | 0 | 100 |
| 20 | 0,31 | 18 | 100 | 8,7 | 0 | 100 | 0,65 | 9,4 | 100 | 6,2 | 0 | 100 |
| 25 | 0,63 | 17 | 100 | 9,6 | 0 | 100 | 1,9 | 12 | 100 | 6 | 0 | 100 |
| 30 | 1,6 | 18 | 100 | 10 | 0 | 100 | 5,8 | 11 | 100 | 6,3 | 0 | 100 |
| Famille 3 | | | | | | | Famille 4 | | | | | |
| | | | | | sans coupes énergétiques | | | | | | sans coupes énergétiques | |
| 10 | 0,064 | 4,1 | 64 | 1,8 | 40 | 100 | 0,037 | 0 | 0,037 | 0 | 100 | 100 |
| 20 | 2,8 | 6,8 | 100 | 3,9 | 0 | 100 | 0,61 | 0,72 | 0,63 | 0 | 100 | 100 |
| 25 | 17 | 4,5 | 100 | 4,5 | 0 | 100 | 1,5 | 0,31 | 1,5 | 0 | 100 | 100 |
| 30 | 24 | 6,4 | 100 | 4,1 | 0 | 80 | 4,8 | 0,07 | 15 | 0 | 89 | 89 |
| | | | | | avec coupes énergétiques à la racine | | | | | | avec coupes énergétiques à la racine | |
| 10 | 0,063 | 3,4 | 80 | 1,8 | 20 | 100 | 0,045 | 1,1 | 0,046 | 0 | 100 | 100 |
| 20 | 2,5 | 6,7 | 100 | 3,8 | 0 | 100 | 0,56 | 0,64 | 0,71 | 0 | 100 | 100 |
| 25 | 18 | 5,8 | 100 | 4,4 | 0 | 100 | 2,6 | 0,072 | 2,9 | 0 | 100 | 100 |
| 30 | 24 | 7,6 | 100 | 4,2 | 0 | 70 | 6,4 | 0 | 6,4 | 0 | 100 | 100 |

TABLE 6.1 – Résultats du PLNE indexé par le temps du CECSP avec et sans coupes énergétiques.

### 6.2.2 Modèles à événements

Cette sous-section, dédiée à la présentation des résultats obtenus pour les modèles à événements, commence par présenter l'algorithme de séparation qui sera utilisé dans le modèle On/Off pour séparer les inégalités de non préemption. Cet algorithme a été conçu en collaboration avec Tamás Kis. Nous présenterons ensuite les différents résultats obtenus pour les modèles Start/End et On/Off.

#### Algorithme de séparation pour les inégalités de non préemption

L'idée principale de la procédure de séparation pour les inégalités de non préemption (voir sous-section 5.2.2) est que, pour chaque activité $i$, trouver la coupe à ajouter au modèle est équivalent à trouver un plus long chemin dans un certain graphe. Ce graphe orienté et acyclique est défini de la manière suivante :

— à chaque événement de l'ensemble $\mathcal{E}^*=\mathcal{E} \setminus \{2n\}$, i.e. $\{1, \ldots, 2n-1\}$, on fait correspondre un sommet ;

— on ajoute au graphe un source et un puits, indexés respectivement par 0 et $2n$. Le graphe est donc composé de $2n+1$ sommets.

— L'ensemble des arcs est divisé en trois catégories :

(i) les *arcs de départ* relient le sommet source 0 à chaque sommet $u \in \{1, \ldots, 2n-3\}$ ;

(ii) les *arcs intermédiaires* relient les sommets $u \in \{1, \ldots, 2n-3\}$ aux sommets $v \in \{u+2, \ldots, 2n-1\}$ et

(iii) les *arcs terminaux* relient les sommets $u \in \{3, \ldots, 2n-1\}$ au sommet puits $2n$ ;

De plus, un coût $cost(u, v)$ est associé à chaque arc $(u, v)$ composant le graphe. Le coût d'un arc de départ est 0, celui d'un arc intermédiaire $(u, v)$ est $cost(u, v) = \overline{z}_{i,u} - \min\{\overline{z}_{i,\ell} \ : \ \ell = u+1, \ldots, v-1\}$ et celui d'un arc terminal $(u, 2n)$ est $\overline{z}_{i,u}$, avec $\overline{z}_{i,u}$ la valeur de la variable correspondante. La construction de ce graphe est illustrée dans l'exemple 6.2.1.

**Exemple 6.2.1.** *Considérons une instance à quatre activités. Le nombre d'événements $|\mathcal{E}|$ est égal à 8. Considérons, par exemple, l'activité 1. Si nous appliquons la transformation décrite ci-dessus, nous obtenons le graphe à 9 sommets et 30 arcs décrits par la figure 6.1. Sur ce graphe, seuls les coûts des arcs terminaux et des arcs intermédiaires sortant du sommet 2 sont représentés.*

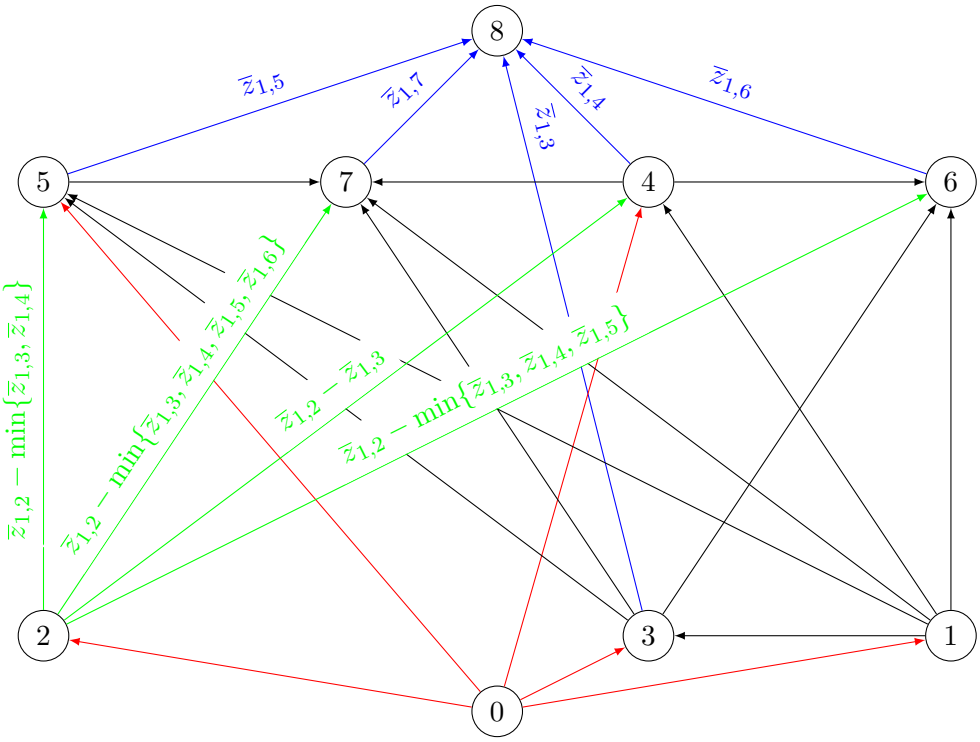



FIGURE 6.1 – Création du graphe de l'algorithme de séparation pour les inégalités de non préemption

Pour séparer un vecteur $\overline{z}_i \in \mathbb{R}^{\mathcal{E}^*}$, nous calculons un plus long chemin dans le graphe à l'aide de la programmation dynamique. Pour cela, nous calculons, pour chaque $v \in \{1, \ldots, 2n-1\}$, la valeur du plus long chemin jusqu'à ce sommet $F(v)$ de la manière suivante :

$$F(v) = \max\{F(u) + cost(u, v) \ : \ u = 1, \ldots, v-2\} \tag{6.1}$$

avec $F(1) = F(2) = 0$.

Puis, on ajoute à ce plus long chemin, la valeur de l'arc servant à rejoindre le puits. Cela revient à ajouter $\overline{z}_{i,v}$ à chaque $F(v)$. Si cette valeur est supérieure à celle du plus long chemin trouvé jusqu'à maintenant, nous remplaçons le plus long chemin et sa valeur par le nouveau chemin trouvé.

**Exemple 6.2.2.** *Considérons le vecteur $\overline{z}_1$ suivant :* $(0,2 \ ; \ 0 \ ; \ 0,4 \ ; \ 0,6 \ ; \ 0,6 \ ; \ 0 \ ; \ 1 \ ; \ 0,2)$.

*Nous avons $F(1) = F(2) = 0$. En effet, l'inégalité devant comporter au moins trois éléments, $F(1)$ et $F(2)$ ne pourront conduire à une telle inégalité. La longueur du plus long chemin est initialisée à 0 et correspond au chemin vide. De plus,nous avons :*

— $F(3) = F(1) + cost(1,3) = F(1) + \overline{z}_{1,1} - \overline{z}_{1,2} = 0 + 0,2 - 0 = 0,2$.
*Si nous calculons* $F(3) + \overline{z}_{1,3} = 0,2 + 0,4 = 0,6$. *La longueur du plus long chemin est donc mise à jour et vaut* $0,6$. *Cette valeur correspond au chemin* $\{1,2,3\}$.

— $$F(4) = \max \begin{cases} F(1) + cost(1,4) & = & F(1) + \overline{z}_{1,1} - \min\{\overline{z}_{1,2}, \overline{z}_{1,3}\} \\ F(2) + cost(2,4) & = & F(2) + \overline{z}_{1,2} - \overline{z}_{1,3} \end{cases}$$
$$= \max \begin{cases} 0 + 0,2 - 0 & = & 0,2 \\ 0 + 0 - 0,4 & = & -0,4 \end{cases}$$
$$= 0,2$$
*Et* $F(4) + \overline{z}_{1,4} = 0,2 + 0,6 = 0,8$. *La longueur du plus long chemin est donc mise à jour et vaut* $0,8$. *Cette valeur correspond au chemin* $\{1,2,4\}$.

— $$F(5) = \max \begin{cases} F(1) + cost(1,5) & = & F(1) + \overline{z}_{1,1} - \min\{\overline{z}_{1,2}, \overline{z}_{1,3}, \overline{z}_{1,4}\} \\ F(2) + cost(2,5) & = & F(2) + \overline{z}_{1,2} - \min\{\overline{z}_{1,3}, \overline{z}_{1,4}\} \\ F(3) + cost(3,5) & = & F(3) + \overline{z}_{1,3} - \overline{z}_{1,4} \end{cases}$$
$$= \max \begin{cases} 0 + 0,2 - 0 & = & 0,2 \\ 0 + 0 - 0,4 & = & -0,4 \\ 0,2 + 0.4 - 0,6 & = & 0 \end{cases}$$
$$= 0,2$$
*Si nous calculons* $F(5) + \overline{z}_{1,5} = 0,2 + 0,6 = 0,8$. *Le plus long chemin n'est donc pas mis à jour.*

— $F(6) = 0.2$ *et* $F(6) + \overline{z}_{1,6} = 0,2 + 0 = 0,2$. *Le plus long chemin n'est pas mis à jour.*

— $F(7) = F(4) + \overline{z}_{1,4} - \min\{\overline{z}_{1,5}, \overline{z}_{1,6}\} = 0,2 + 0,6 - 0 = 0,8$. *De plus,* $F(7) + \overline{z}_{1,7} = 0,8 + 1 = 1,8$. *La longueur du plus long est donc mise à jour, vaut* $1,8$ *et correspond au chemin* $\{1,2,4,6,7\}$

*A la fin de la procédure, le plus long chemin trouvé est donc* $\{1,2,4,6,7\}$ *et l'inégalité correspondante, i.e. celle qui sera ajoutée au modèle, est :*

$$z_{1,1} - z_{1,2} + z_{1,4} - z_{1,6} + z_{1,7} \leq 1$$

**Modèles Start/End**

Dans ce paragraphe, nous présentons les résultats obtenus pour le modèle Start/End. Les expérimentations ont été conduites sur les instances de la Famille 1 présentée dans la sous-section 6.1.1 et avec une limite de temps de 1000 secondes. Le modèle est utilisé avec l'objectif de minimisation de la consommation totale de ressource. Le tableau 6.2 présente ces résultats.

Dans ce tableau, nous avons comparé la qualité et le temps d'obtention des premières solutions pour chaque famille d'instances testées. Nous avons aussi comparé le temps nécessaire à l'obtention de la solution optimale (7200 secondes si la solution trouvée n'est pas optimale), la qualité de la solution, le nombre d'instances résolues et le nombre d'entre elles résolues à l'optimum.

Dans le tableau 6.2, nous pouvons voir que le modèle Start/End permet de résoudre seulement une petite partie des instances à l'optimum. Une solution est cependant trouvée pour toutes les instances mais la qualité de ces dernières n'est pas très bonne. De meilleures solutions, obtenues avec le modèle On/Off, sont présentées dans le paragraphe suivant.

| #act. | $1^{ère}$ sol. | | sol. finale | | | |
|---|---|---|---|---|---|---|
| | tps(s) | gap | tps | gap | %solv. | %opt. |
| 10 | 0,42 | 95 | 5200 | 42 | 100 | 33 |
| 20 | 73 | 80 | 7200 | 29 | 100 | 0 |
| 25 | 330 | 94 | 7200 | 58 | 100 | 0 |
| 30 | 200 | 82 | 6500 | 48 | 100 | 0 |
| 60 | 310 | 140 | 7200 | 63 | 100 | 0 |

TABLE 6.2 – Résultats du modèle Start/End pour le CECSP(Famille 1).

### Modèles On/Off

**Résultats du modèle On/Off pour le CECSP** Dans ce paragraphe, nous présentons les résultats obtenus pour le modèle On/Off. Les expérimentations conduites pour ce modèle dans le cadre du CECSP l'ont été sur les instances de la Famille 4 (cf. tableau 6.3) et sur les instances de la Famille 1 (cf. tableau 6.4) avec une limite de temps de 1000 secondes.

Pour la Famille 4, le tableau 6.3 présente le pourcentage d'instances résolues à l'optimum ainsi que le temps nécessaire à leur résolution pour différentes combinaisons de coupes ajoutées au modèle. Dans le tableau, ces différentes inégalités sont représentées de la manière suivante :

- *Sep.* ou *S.* représente les inégalités bornant supérieurement la distance entre deux événements ;
- *Date* ou *D.* les inégalités bornant supérieurement la date des événements ;
- *KP* les inégalités déduites du problème du sac-à-dos, et
- $\overline{Preem.}$ ou $\overline{P.}$ les inégalités de non préemption.

Ces inégalités sont décrites dans la sous-section 5.2.2. Les deux premiers ensembles d'inégalités sont ajoutés directement dans le modèle et sont aussi utilisées comme borne plus fine dans les contraintes du modèle (voir sous-section 5.2.2).

| #act. / ineg. | 10 | | 20 | | 25 | | 30 | |
|---|---|---|---|---|---|---|---|---|
| | tps(s) | %opt | tps(s) | %opt | tps(s) | %opt | tps(s) | %opt |
| *Aucune* | 0,3 | 100 | 164,14 | 90,9 | 635,4 | 55,6 | 968 | 10 |
| *Sep.* | 0,6 | 100 | 182,2 | 90,9 | 727,3 | 55,6 | 851 | 20 |
| *Date* | 0,5 | 100 | 167,3 | 90,9 | 629,5 | 88,9 | 961,4 | 20 |
| *KP* | 0,5 | 100 | 164,7 | 90,9 | 555,1 | 66,7 | 845,3 | 20 |
| $\overline{Preem.}$ | 0,3 | 100 | 330,9 | 72,7 | 822,4 | 44,4 | 914,8 | 10 |
| *S.* & *D.* & *KP* | 0,6 | 100 | 154 | 90,9 | 389,9 | 77,8 | 839,9 | 20 |
| *S.* & *D.* & $\overline{P.}$ | 0,6 | 100 | 179,8 | 90,9 | 454,4 | 77,8 | 813,8 | 60 |
| *S.* & *KP* & $\overline{P.}$ | 0,5 | 100 | 182,2 | 90,9 | 759,6 | 33,3 | 924,9 | 20 |
| *D.* & *KP* & $\overline{P.}$ | 0,5 | 100 | 170 | 90,9 | 705 | 66,7 | 816 | 50 |
| *S.* & *D.* & *KP* & $\overline{P.}$ | 0,9 | 100 | 278,9 | 81,8 | 510 | 88,9 | 802,8 | 50 |

TABLE 6.3 – Résultats du modèle On/Off pour le CECSP avec différentes combinaisons de coupes (Famille 4).

Dans le tableau 6.3, nous pouvons remarquer que le nombre d'instances à 10 et 20 activités résolues est du même ordre de grandeur pour toutes les combinaisons d'inégalités testées. Pour les instances à 25 activités, les résultats sont plus hétérogènes mais les meilleurs résultats sont obtenus en combinant toutes les inégalités. Enfin, pour les instances à 30 activités, les meilleurs résultats sont quant à

| #act. / ineg. | 10 | | 20 | | 25 | | 30 | |
|---|---|---|---|---|---|---|---|---|
| | %feas. | gap | %feas. | gap | %feas. | gap | %feas. | gap |
| *Aucune* | 100 | 23,2 | 81,8 | 74,5 | 22,2 | 91,9 | 0 | 0 |
| *Sep* | 100 | < 0,01 | 100 | 14,9 | 44,4 | 68,9 | 0 | 0 |
| *Date* | 100 | 1,06 | 100 | 70,5 | 33,3 | 88,1 | 40 | 84,9 |
| *KP* | 100 | 9,8 | 100 | 46,4 | 77,78 | 64,6 | 25 | 70,7 |
| $\overline{Preem.}$ | 100 | 61,5 | 81,8 | 73,4 | 0 | 0 | 0 | 0 |
| *S. & D. & KP* | 100 | < 0,01 | 100 | 13,7 | 100 | 51,5 | 50 | 70,6 |
| *S. & D. &* $\overline{P.}$ | 100 | 4,77 | 100 | 45,5 | 88,89 | 75,1 | 30 | 84,6 |
| *S. & KP &* $\overline{P.}$ | 100 | 8,67 | 100 | 68,3 | 44,4 | 88,1 | 10 | 93 |
| *D. & KP &* $\overline{P.}$ | 100 | 23,8 | 100 | 60,1 | 83,3 | 79,8 | 30 | 88,2 |
| *S. & D. & KP &* $\overline{P.}$ | 100 | 0,07 | 100 | 35,7 | 100 | 66,9 | 10 | 87,1 |

TABLE 6.4 – Résultats du modèle On/Off pour le CECSP avec différentes combinaisons de coupes (Famille 1).

eux obtenus en combinant seulement inégalités de non préemption et les deux ensembles d'inégalités portant sur les dates des événements, i.e. *Sep* et *Date*. Cependant, les résultats obtenus en combinant toutes les inégalités ne sont pas très éloignés de ceux obtenant les meilleurs résultats.

Pour les instances de la Famille 1, seules un petit nombre d'instances sont résolues de manière optimale. C'est pourquoi nous présentons uniquement le pourcentage d'instances pour lesquelles une solution a été trouvée et nous calculons la distance entre cette solution et la meilleure borne inférieure trouvée à la fin du temps imparti. Ces résultats sont présentés dans le tableau 6.4. Les notations sont les mêmes que celles utilisées dans le tableau 6.3.

Dans le tableau 6.3, nous pouvons voir que les meilleurs résultats sont obtenus en combinant *Sep, Date* et *KP*. Cependant, des résultats comparables sont obtenus en combinant *Sep, Date* et $\overline{Preem.}$ ou *Sep, Date, KP* et $\overline{Preem.}$.

**Résultats du modèle On/Off pour le RCPSP** Nous allons maintenant évaluer les performances relatives des améliorations du modèle On/Off dans le cadre du RCPSP. Pour ce faire, les expérimentations ont été conduites sur les instances de Koné *et al.* [KALM11] et décrites dans la sous-section 6.1.2 avec un pré-calcul des fenêtres de temps des activités aussi décrit dans la sous-section 6.1.2. La limite de temps est fixée à 1000 secondes.

Pour ces instances, le tableau 6.5 présente le pourcentage d'instances résolues à l'optimum ainsi que le temps nécessaire à leur résolution pour différentes combinaisons de coupes ajoutées au modèle.

| #act. / ineg. | $1^{ère}$ sol. | | Sol. finale | | |
|---|---|---|---|---|---|
| | tps(s) | gap | tps(s) | gap | %opt. |
| *Aucune* | 0,19 | 3,4 | 34,8 | 0 | 100 |
| *Sep.* | 0,17 | 3,2 | 30,3 | 0 | 100 |
| $\overline{Preem.}$ | 0,17 | 3,8 | 30,3 | 0 | 100 |
| *Sep. &* $\overline{Preem.}$ | 0,29 | 3,3 | 33,1 | 0 | 100 |

TABLE 6.5 – Résultats du modèle On/Off pour le RCPSP avec différentes combinaisons de coupes.

Dans le tableau 6.5, nous pouvons voir que les résultats de l'ajout des coupes et inégalités a moins d'impact dans le cadre du RCPSP. Cependant, une amélioration des performances du modèle, spécialement pour l'ajout des inégalités *Sep.* ou $\overline{Preem.}$, peut être notée. Cette amélioration est moindre dans le cas où les deux ensembles d'inégalités sont ajoutés simultanément.

### 6.2.3 Comparaison des différentes approches

Dans cette sous-section, nous allons effectuer des comparaisons entre les différentes méthodes décrites ci-dessus. Dans un premier temps et pour montrer que, dans certains cas, le modèle indexé par le temps peut produire des solutions sous-optimales, nous comparons les résultats de ce dernier avec ceux du modèle On/Off (sans ajout de coupes particulières) sur la Famille d'instances 1.

Le tableau 6.6 présente ces résultats. Pour chacune des formulations, la première colonne décrit le temps passé dans la résolution du PLNE. La seconde colonne représente le pourcentage d'instances résolues à l'optimum et la troisième colonne le nombre d'instances pour lesquelles une solution réalisable a été trouvé. Enfin, la dernière colonne du tableau montre la différence moyenne entre la valeur de l'objectif retourné par les deux modèles. Par exemple, la première valeur de cette colonne nous dit que, pour les instances à 20 activités, la valeur de l'objectif retourné par le modèle indexé par le temps est, en moyenne, 10% plus élevée que celle retournée par le modèle à événements.

| | Modèle On/Off | | | Modèle indexé par le temps | | | |
|---|---|---|---|---|---|---|---|
| #act. | tps(s) | %opt. | %feas. | tps(s) | %opt. | %feas. | %dev. obj. |
| 20 | 7200 | 0 | 100 | 7200 | 0 | 100 | 10,08 |
| 25 | 7200 | 0 | 44,44 | 7200 | 0 | 66,67 | 10,11 |

TABLE 6.6 – Comparaison du modèle On/Off et du modèle indexé par le temps pour le CECSP.

Tout d'abord, nous pouvons remarquer que le modèle indexé par le temps permet de trouver une solution réalisable pour plus d'instances que le modèle On/Off. Cependant, aucun des deux modèles n'est capable de résoudre les instances de manière optimale. Enfin, le modèle On/Off trouve clairement de meilleures solutions que le modèle indexé par le temps ce qui tend à montrer que la considération du temps continu est profitable en pratique pour économiser les ressources utilisées.

Les modèles présentés dans ce manuscrit ont de grandes difficultés à trouver des solutions optimales. En particulier lorsque des fonctions de rendement, même seulement affines, entrent en jeu. De plus, les méthodes testées dans la section suivante, portant sur les techniques issues de la programmation par contraintes, ne le sont que pour la variante décisionnelle du CECSP, i.e. sans objectif. De ce fait, nous allons présenter les résultats des trois différents modèles dans le cas où aucun objectif n'est présent. Le tableau 6.7 présente ces résultats.

Pour chacun des modèles, trois colonnes exposent les résultats. La première correspond au temps nécessaire pour trouver une solution (si une solution est trouvée). La seconde correspond au pourcentage d'instance résolue et la dernière colonne correspond au nombre d'instances prouvées réalisables. Cette dernière colonne est introduite pour permettre de montrer que le modèle indexé par le temps peut prouver l'infaisabilité d'une instance alors que les modèles a événements trouvent une solution pour cette même instance prouvant qu'elle est en fait réalisable. De ce fait, le modèle indexé par le temps sur-contraint bien de manière significative le problème initial.

| | Indexé par le temps | | | On/Off | | | Start/End | | |
|---|---|---|---|---|---|---|---|---|---|
| #act. | tps(s) | %solv. | %feas. | tps(s) | %solv. | %feas. | tps(s) | %solv. | %feas. |
| Famille 1 | | | | | | | | | |
| 10 | 0.03 | 100 | 60 | 0,22 | 100 | 100 | 0,71 | 100 | 100 |
| 20 | 0,08 | 100 | 54,5 | 11,57 | 100 | 100 | 355,4 | 100 | 100 |
| 25 | 0,22 | 100 | 66,7 | 58,79 | 100 | 100 | 2226,7 | 77,8 | 77,8 |
| 30 | 0,25 | 100 | 60 | 1582,9 | 80 | 80 | 6247,2 | 20 | 20 |
| 60 | 420 | 100 | 100 | 4969,51 | 80 | 80 | 6219,8 | 20 | 20 |
| Famille 3 | | | | | | | | | |
| 10 | 0,04 | 100 | 100 | 0,46 | 100 | 100 | 0,88 | 100 | 100 |
| 20 | 0,24 | 100 | 100 | 658,362 | 100 | 100 | 1430,6 | 90,9 | 90,9 |
| 25 | 0,45 | 100 | 88,9 | 1900,17 | 77,78 | 77,78 | 5816,1 | 33,3 | 33,3 |
| 30 | 1,38 | 100 | 100 | 2819,7 | 80 | 80 | 6011,8 | 20 | 20 |
| 60 | 328,7 | 100 | 80 | 6358,15 | 20 | 20 | 7200 | 0 | 0 |
| Famille 4 | | | | | | | | | |
| 10 | 0,01 | 100 | 100 | 0,23 | 100 | 100 | 0,7 | 100 | 100 |
| 20 | 0,27 | 100 | 100 | 734,04 | 90,9 | 90,9 | 2995,9 | 63,6 | 63,6 |
| 25 | 0,7 | 100 | 100 | 2102,85 | 77,8 | 77,8 | 4833,6 | 44,4 | 44,4 |
| 30 | 1,57 | 100 | 100 | 4483,4 | 60 | 60 | 6485,9 | 20 | 20 |
| 60 | 224,72 | 100 | 60 | 7200 | 0 | 0 | 7200 | 0 | 0 |

TABLE 6.7 – Comparaison des trois modèles de PLNE du CECSP sans fonction objectif.

Comme nous pouvons le voir dans le tableau 6.7, le modèle indexé permet de trouver des solutions beaucoup plus rapidement que les modèles à événements. Cependant, ce modèle ne peut être employé comme une méthode de résolution exacte puisque, pour certaines instances, seules des solutions fractionnaires existent. Ces résultats montrent ainsi encore que la résolution en temps continu peut apporter des gains significatifs en pratique.

Parmi les deux formulations basées sur les événements, la formulation On/Off est celle qui fournit les meilleurs résultats en termes de temps de calcul et de nombre d'instances résolues.

## 6.3 Performances de la Programmation Par Contraintes

Dans cette section, nous présentons les résultats des expérimentations portant sur les méthodes présentées dans le chapitre 3. Dans un premier temps, nous définissons le cadre des expérimentations, notamment les algorithmes et heuristiques de choix de variables utilisées. Puis, dans un second temps, nous présenterons en détail les résultats numériques issus de ces expérimentations.

### 6.3.1 Cadre des expérimentations

Pour mesurer les performances relatives des différents raisonnements présentés dans le chapitre 3, nous les intégrons dans une procédure de branchement hybride [NAL15b, NALR16]. Cette procédure se divise en deux temps. Dans un premier temps, une méthode arborescente est utilisée afin de réduire la taille des domaines des début et fin des activités jusqu'à ce que chaque domaine ait une taille inférieure à un certain paramètre $\epsilon > 0$. Une fois les domaines réduits suffisamment, nous utilisons le modèle à événements On/Off afin de fixer les dates de début et de fin et de calculer, pour chaque activité $i$, sa fonction d'allocation de ressource $b_i(t)$. Nous rappelons que, pour ce faire, seule la valeur

de $b_i(t)$ à chaque début et fin d'activité doit être calculée (conséquence du théorème 1.2).

La procédure de branchement est inspirée du travail de Carlier *et al.* [CL91]. Au début de cette procédure, une activité peut commencer (respectivement finir) à tout instant $t \in [est_i, lst_i]$ (resp. $t \in [eet_i, let_i]$). L'idée de l'algorithme est donc, à chaque nœud, de réduire la taille d'un de ces intervalles. Par exemple, supposons que l'on ait choisi de réduire la taille du domaine de $st_i$, alors deux nouveaux nœuds sont créés : le premier avec la contrainte supplémentaire $st_i \in [est_i, (est_i + lst_i)/2]$ et le second avec la contrainte $st_i \in ](est_i + lst_i)/2, lst_i]$ (voir figure 6.2) ; et à chaque nœud, un des raisonnements présentés au chapitre 3 est appliqué.

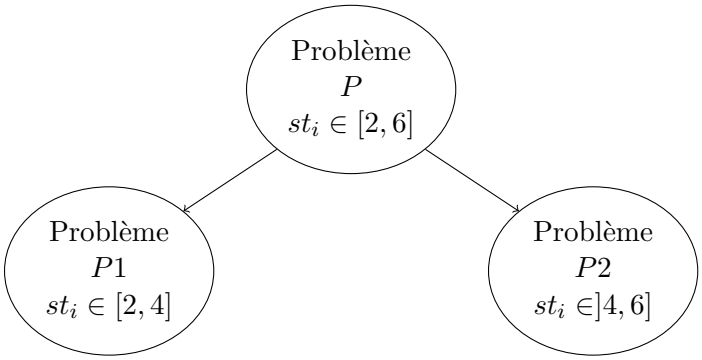



FIGURE 6.2 – Procédure de branchement du l'algorithme hybride du CECSP.

Cette procédure est répétée jusqu'à ce que tous les domaines de toutes les variables soient plus petits qu'un certain paramètre $\epsilon > 0$. Quand ceci arrive, cela veut dire que nous sommes au niveau d'une feuille de notre arbre de recherche et nous pouvons évaluer la satisfiabilité du nœud courant. Pour cela, nous utilisons le modèle On/Off, le plus efficace dans le cadre d'une résolution exacte, pour tester si une solution avec les contraintes supplémentaires définies le long de la procédure de branchement existe.

Si une telle solution existe alors l'algorithme s'arrête et nous avons trouvé une solution pour l'instance du CECSP testée. Sinon, nous faisons marche arrière dans l'arbre de recherche afin d'évaluer d'autres feuilles de l'arbre, i.e. d'autres solutions potentielles.

Le parcours de l'arbre est fait en suivant un parcours en profondeur et, à chaque nœud, la variable dont on va réduire le domaine est choisie selon l'heuristique choisissant la variable de plus petit domaine. De plus, la taille moyenne des domaines des variables étant de 32, les valeurs de $\epsilon$ testées sont 10, 5, 2.5.

### 6.3.2 Raisonnement énergétique

Dans un premier temps, nous allons commencer par comparer les différentes méthodes de calcul des intervalles d'intérêt pour l'algorithme de vérification de ce raisonnement, présentées dans la sous-section 3.2.3.

**Comparaison des méthodes de calcul des intervalles d'intérêt pour l'algorithme de vérification du raisonnement énergétique**

Les résultats comparant les trois méthodes de calcul des intervalles d'intérêt de l'algorithme de vérification du raisonnement énergétique sont présentés dans le tableau 6.8. La première colonne correspond à l'algorithme naïf de calcul d'intersection des segments de définition des fonctions de consommation individuelle des activités. La seconde colonne correspond quant à elle au calcul de ces mêmes intersections à l'aide de l'algorithme de balayage de Bentley-Ottmann. Enfin, la troisième colonne présente les résultats de l'adaptation de la méthode de calcul de Derrien *et al.* [DP14] pour le problème cumulatif dans le cadre du CECSP. Toutes ces méthodes sont décrites dans la sous-section 3.2.3.

L'algorithme de balayage fait partie de la librairie C++ CGAL [1]. Pour calculer ces performances, nous appliquons l'algorithme de vérification du raisonnement énergétique et les ajustements correspondant sur les intervalles de l'algorithme de vérification seulement. Cet algorithme est appliqué sur toutes les instances des Familles 1, 2, 3 et 4. Le temps est représenté en millisecondes.

| # act. | méthode naïve | algorithme de balayage | adaptation de l'algorithme de [DP14] |
|---|---|---|---|
| 10 | 0,46 | 1,57 | 0,39 |
| 20 | 3,9 | 6,2 | 1,05 |
| 25 | 7,18 | 7,50 | 1,73 |
| 30 | 11,54 | 11,78 | 3,06 |
| 60 | 45,97 | 62,82 | 14,40 |

TABLE 6.8 – Comparaison des méthodes de calcul des intervalles d'intéret du raisonnement énergétique.

La meilleure méthode permettant le calcul des intervalles d'intérêt de l'algorithme de vérification du raisonnement énergétique est la méthode adaptée de [DP14]. En effet, un des avantages de cette méthode était que le nombre d'intervalles à calculer était beaucoup plus faible que dans les autres cas. Il paraît donc cohérent que cet algorithme ait les meilleures performances.

Cependant, il est moins naturel que l'algorithme par balayage ait de moins bonnes performances que la méthode naïve. Ceci est dû au fait que la complexité de l'algorithme de balayage dépend du nombre d'intersections que l'algorithme doit calculer. Or, dans ce cas, ce nombre est très grand, ce qui ralentit grandement le temps de calcul de l'algorithme.

Dans le reste des expérimentations conduites sur le raisonnement énergétique, nous utiliserons donc la méthode de calcul adaptée de [DP14].

**Intégration du raisonnement énergétique dans l'algorithme de branchement hybride**

Dans ce paragraphe, nous présentons les résultats des expérimentations faites pour le raisonnement énergétique. Ce raisonnement est intégré dans l'algorithme de branchement hybride décrit dans la sous-section 6.3.1. Cet algorithme a été testé avec l'heuristique de sélection de variable qui choisit la variable de plus petit domaine.

1. CGAL, Computational Geometry Algorithms Library. http ://www.cgal.org.

Le tableau 6.9 présente les résultats obtenus avec pour valeur de $\epsilon = 2,5$ . Cette valeur est choisie car c'est celle qui permettait d'obtenir les meilleurs résultats. Les autres valeurs de paramètres testées étant : 10 et 5. Dans ce tableau, la première colonne décrit le temps nécessaire pour résoudre les instances. Les deuxième et troisième colonnes montrent le pourcentage de temps passé dans la résolution du PLNE et dans l'arbre de recherche respectivement. La quatrième colonne énonce le pourcentage d'instances résolues. Les cinquième et sixième colonnes décrivent respectivement le nombre de noeuds contenus dans l'arbre de recherche et de programmes linéaires en nombres entiers résolus. Enfin, la dernière colonne exhibe le nombre d'ajustements appliqués.

| #act. | Méthode de branchement hybride $\epsilon = 2.5$ | | | | | | |
|---|---|---|---|---|---|---|---|
| Famille 1 | | | | | | | |
| | Tps total(s) | Tps CPLEX(%) | Tps arbre(%) | %solv. | #noeuds | #PLNE | #adj. |
| 10 | 0,09 | 83,19 | 16,81 | 100 | 25 | 6 | 6 |
| 20 | 660,21 | 96,97 | 3,03 | 90 | 3809 | 20611 | 2716 |
| 25 | 821,9 | 98,3 | 1,7 | 88 | 89 | 10 | 95 |
| 30 | 112,58 | 99,14 | 0,86 | 100 | 102 | 10 | 114 |
| Famille 2 | | | | | | | |
| | Tps total(s) | Tps CPLEX(%) | Tps arbre(%) | %solv. | #noeuds | #PLNE | #adj. |
| 10 | 1109,5 | 81,2 | 18,75 | 100 | 46783 | 140276 | 65446 |
| 20 | 728,04 | 97,55 | 2,45 | 90 | 1930 | 9296 | 3851 |
| 25 | 1825,15 | 98,7 | 1,3 | 75 | 495 | 761 | 458 |
| 30 | 1971,81 | 99,62 | 0,38 | 75 | 150 | 5 | 659 |
| Famille 4 | | | | | | | |
| | Tps total(s) | Tps CPLEX(%) | Tps arbre(%) | %solv. | #noeuds | #PLNE | #adj. |
| 10 | 0,19 | 87,55 | 12,45 | 100 | 25 | 5 | 33 |
| 20 | 1617,07 | 98,97 | 1,03 | 77 | 68 | 9 | 246 |
| 25 | 104,9 | 99,6 | 0,4 | 100 | 66 | 6 | 252 |
| 30 | 1749,76 | 99,72 | 0,27 | 77 | 107 | 9 | 424 |

TABLE 6.9 – Résultats du raisonnement énergétique dans la méthode de branchement hybride pour le CECSP.

Tout d'abord, nous pouvons remarquer que l'algorithme de branchement hybride permet de résoudre des instances jusqu'à 30 activités en moins de 2000 secondes. De plus, nous pouvons voir qu'une bonne partie du temps de résolution est passé dans le programme linéaire mixte. Enfin, le raisonnement énergétique permet de procéder à un nombre important d'ajustements.

### 6.3.3 Raisonnements basés sur le Time-Table

Dans cette sous-section, nous montrons l'intérêt d'ajouter le raisonnement basé sur un problème de flot et sur le raisonnement Time-Table. Pour ce faire, à chaque noeuds de l'arbre de recherche, nous appliquons l'algorithme de Time-Table/Flot pour détecter une incohérence.

Pour ces expérimentations, nous avons aussi considéré l'heuristique choisissant la variable de plus

petit domaine. Le tableau 6.10 présente les résultats obtenus avec pour valeur de $\epsilon = 2,5$. Les colonnes du tableau correspondent à celle du tableau 6.9.

| #act. | Méthode de branchement hybride $\epsilon = 2.5$ | | | | | |
|---|---|---|---|---|---|---|
| Famille 1 | | | | | | |
| | Tps total(s) | Tps CPLEX(%) | Tps arbre(%) | %solv. | #noeuds | #PLNE |
| 10 | 0,12 | 51,5 | 48,5 | 100 | 25 | 6 |
| 20 | 5,0 | 88,7 | 11,3 | 100 | 58 | 12 |
| 25 | 26,6 | 85,8 | 14,2 | 100 | 79 | 10 |
| 30 | 70,8 | 96,69 | 3,3 | 100 | 100 | 11 |
| Famille 2 | | | | | | |
| | Tps total(s) | Tps CPLEX(%) | Tps arbre(%) | %solv. | #noeuds | #PLNE |
| 10 | 0,17 | 63,0 | 37,0 | 100 | 24 | 6 |
| 20 | 9,03 | 86,5 | 13,5 | 100 | 59 | 12 |
| 25 | 81,7 | 94,1 | 5,9 | 100 | 86 | 10 |
| 30 | 118,9 | 96,7 | 3,3 | 100 | 106 | 11 |
| Famille 4 | | | | | | |
| | Tps total(s) | Tps CPLEX(%) | Tps arbre(%) | %solv. | #noeuds | #PLNE |
| 10 | 0,29 | 77,3 | 22,7 | 100 | 27 | 6 |
| 20 | 1476,1 | 93,2 | 6,8 | 80 | 98 | 10 |
| 25 | 2568,5 | 90,2 | 9,8 | 66 | 18973 | 8 |
| 30 | 3181,3 | 98,3 | 1,7 | 60 | 20908 | 8 |

TABLE 6.10 – Résultats du Time-Table basé sur les flots dans la méthode de branchement hybride pour le CECSP.

L'algorithme de branchement, combiné au raisonnement basé sur les flots, permet de résoudre des instances du CECSP avec 30 activités en moins de 7200 secondes. De plus, nous pouvons voir que, comme dans le cas du raisonnement énergétique, une bonne partie du temps de résolution est aussi passé dans le programme linéaire mixte. Le raisonnement basé sur les flots permet aussi d'obtenir de meilleurs résultats que le raisonnement énergétique pour les Familles 2 et 1 et de moins bons résultats pour la Famille 4. La recherche de propriétés inhérentes au problème permettant de savoir quel raisonnement appliquer est une direction de recherche intéressante.

### 6.3.4 Comparaison des différentes approches

Dans cette sous-section, nous allons comparer les résultats obtenus avec le modèle On/Off avec ceux obtenus par la méthode de branchement hybride. Ces résultats sont décrits dans le tableau 6.11.

Dans ce tableau, les deux premières colonnes présentent les résultats obtenus par le modèle On/Off et les deux suivantes ceux obtenus par la méthode de branchement hybride.

Dans le tableau 6.11, nous pouvons voir que, pour les Familles 1 et 2, l'algorithme de branchement hybride permet de résoudre plus d'instances que le modèle On/Off en un temps moins important. Dans certains cas, la résolution devient 20 fois plus rapide grâce à l'utilisation de cette méthode.

| #act. | Modèle On/Off | | Algo. hybride | |
|---|---|---|---|---|
| | tps(s) | %solv. | tps | %solv. |
| Famille 1 | | | | |
| 10 | 0, 22 | 100 | 0, 12 | 100 |
| 20 | 11, 56 | 100 | 5, 0 | 100 |
| 25 | 58, 79 | 100 | 26, 6 | 100 |
| 30 | 1582, 9 | 80 | 70, 8 | 100 |
| Famille 2 | | | | |
| 10 | 0, 29 | 100 | 0, 17 | 100 |
| 20 | 27, 2 | 100 | 9, 03 | 100 |
| 25 | 380, 20 | 100 | 81, 7 | 100 |
| 30 | 2634, 3 | 70 | 118, 9 | 100 |

TABLE 6.11 – Comparaison des résultats du modèle On/Off et de ceux de la méthode de branchement hybride pour le CECSP.

Nous allons maintenant présenter les quelques résultats obtenus pour les instances de la famille $LP$ (avec des fonctions de rendement concaves et affines par morceaux). Cependant, comme seules les instances à 10 activités ont pu être résolues, nous présentons uniquement ces résultats. Le tableau 6.12 présente ces résultats avec une limite de temps fixée à 3600 secondes et $\epsilon = 5$. La première ligne du tableau décrit les résultats de la méthode arborescente hybride avec le raisonnement basé sur les flots, la seconde avec le raisonnement énergétique et la dernière avec les deux raisonnements combinés. Pour chaque ligne, la première colonne correspond au temps nécessaire à la résolution des instances. La seconde exhibe le temps passé dans l'arbre de branchement. Enfin, la troisième et quatrième colonne présentent respectivement le nombre de PLNE résolus et le nombre de noeuds dans l'arbre.

| | Tps Total (s) | Tps arbre (s) | #PLNE | #noeuds |
|---|---|---|---|---|
| TTFlot | 3100,8 | 0,01 | 1 | 8 |
| RE | 2315,9 | 0,07 | 14,8 | 40 |
| RE + TTFlot | 1709,9 | 0,14 | 9,25 | 37,75 |

TABLE 6.12 – Résultats de la méthode de branchement arborescente sur les instances avec fonctions de rendement concaves et affines par morceaux.

Dans le tableau 6.12, nous pouvons voir que le raisonnement basé sur l'algorithme de flot est moins efficace lorsqu'il est utilisé sans le raisonnement énergétique mais lorsque les deux raisonnements sont combinés ils deviennent plus efficace que l'un ou l'autre utilisé seul.

Nous pouvons également remarquer est que, lors de la résolution, la plupart du temps est passé dans la recherche d'une solution au PLNE. De ce fait, l'amélioration des performances de cette méthode devra obligatoirement passer par une amélioration des performances du PLNE.

Pour les instances de la Famille $L$, une solution a été trouvée pour seulement 75% des instances. Donc, quand on compare ces résultats avec ceux obtenus pour la Famille $LP$, on peut conclure que, dans un cas sur quatre, approximer la fonction de rendement par une fonction affine rend l'instance infaisable. Ceci permet, malgré la difficulté de résolution de ces instances, de justifier la considération de fonctions de rendement concaves et affines par morceaux.

# Conclusion

Dans cette partie, nous avons présenté les résultats obtenus par les expérimentations que nous avons conduites. Ces expérimentations ont permis de valider et de comparer les méthodes présentées dans ce manuscrit.

Dans un premier temps, nous avons présenté la procédure utilisée pour générer des instances hétérogènes du CECSP et les caractéristiques des instances utilisées dans les expérimentations sur le RCPSP. Pour ce dernier, nous expliquons aussi comment sont précalculées les fenêtres de temps dans lesquelles les activités doivent s'exécuter.

Nous présentons ensuite les résultats obtenus par le modèle indexé par le temps du CECSP. Une comparaison des résultats avec et sans coupes énergétiques est réalisée montrant l'intérêt de ces coupes. En effet, même si l'ajout de ces coupes augmente le temps de résolution du modèle, cela permet de trouver une première solution de meilleure qualité que pour le modèle sans ces coupes. En effet, le nombre important d'inégalités ajoutées au modèle rend la résolution plus délicate mais ces dernières rendent le modèle plus fort. La mise en place d'un algorithme de séparation permettant d'ajouter seulement une partie de ces inégalités à chaque nœud de l'arbre de branchement est donc une piste de recherche intéressante.

La sous-section suivante détaille les résultats obtenus par les modèles à événements. Nous avons présenté, dans un premier temps, un algorithme polynomial utilisé pour séparer les inégalités de non-préemption définies dans la sous-section 5.2.2. Ensuite, les résultats obtenus par le modèle Start/End pour résoudre le CECSP sont détaillés. Ce dernier, bien qu'il ait de meilleures relaxations que le modèle On/Off, ne permet pas de résoudre autant d'instances car il comprend deux fois plus de variables binaires.

Les expérimentations conduites sur le modèle On/Off pour résoudre le CECSP ont permis de valider les ensembles d'inégalités et de coupes définies dans la section 5.2. En effet, plusieurs sous-ensembles de ces inégalités ont successivement été ajoutées au modèle et une comparaison des résultats a été effectuée, montrant que les meilleures performances étaient obtenues quand ces inégalités étaient ajoutées au modèle. De plus, l'intérêt de l'ajout d'une partie de ces inégalités dans le cadre de la résolution du RCPSP a aussi été démontré.

Enfin, une comparaison des résultats obtenus pour les trois modèles a été effectuée dans le paragraphe suivant permettant de vérifier expérimentalement que le modèle indexé par le temps ne peut pas être utilisé comme une méthode de résolution exacte pour le CECSP. De plus, cette comparaison a de nouveau permis de montrer que les résultats obtenus avec le modèle On/Off était meilleurs que ceux obtenus par le modèle Start/End.

La section suivante a été consacré aux résultats obtenus par les algorithmes de filtrage pour le CECSP. Ces derniers sont inclus dans une méthode de branchement hybride, couplant règle de branchement et modèle On/Off. Les expérimentations conduites ont permis de montrer que cet algorithme obtenait de meilleurs résultats que le modèle On/Off seul. De plus, l'intérêt et la complémentarité de l'algorithme du raisonnement énergétique et celui de l'algorithme de vérification basé sur les flots présentés dans le chapitre 5 ont été montrés. En effet, pour certaines familles d'instances l'algorithme de flot obtient de meilleurs résultats et, pour d'autres familles, c'est le raisonnement énergétique qui les obtient.

Enfin, cette partie a comparé les deux algorithmes de filtrage sur des instances comportant des fonctions de rendement concaves et affines par morceaux. Pour ces instances, difficiles à résoudre, nous appliquons l'algorithme de branchement hybride avec un des deux raisonnements puis avec les deux. Ceci nous permet de démontrer que, dans ce cas, l'algorithme de vérification basé sur les flots obtient de meilleures performances lorsqu'il est couplé avec le raisonnement énergétique. La dernière information que ces expérimentations nous permettent d'obtenir est que l'approximation des fonctions de rendement par des fonctions affines peut conduire à l'infaisabilité de l'instance alors que l'approximation par des fonctions concaves et affines par morceaux permet de trouver une solution. Ce dernier point permet, malgré la difficulté de résolution de ces instances, de justifier la considération de fonctions de rendement concaves et affines par morceaux.

# Conclusions et Perspectives

Beaucoup de problèmes d'ordonnancement cumulatifs définis dans la littérature souffrent d'une limitation imposant que chaque activité ait une durée et une consommation de ressource fixes durant son exécution. Il arrive cependant, dans de nombreux cas pratiques, que cette limitation empêche la modélisation correcte du problème. De ce fait, de nouveaux problèmes permettant de modéliser la malléabilité des activités, i.e. ayant une durée et une consommation de ressource variables, ont été introduits [DRDH98, NK14, FT10, Kis05, BLPN99]. Cette thèse introduit un nouveau problème appartenant à cette classe, le CECSP. Une comparaison des différents problèmes existant permettant de modéliser des activités malléables a été réalisée et a permis de montrer que le CECSP est très différents de ces derniers.

La principale difficulté du CECSP repose sur la combinaison entre ressource continue et malléabilité des activités. En effet, ces deux caractéristiques impliquent que les activités peuvent prendre des formes quasi quelconques. De ce fait, un des premiers travaux effectués dans cette thèse a été une étude détaillée du problème. Cette étude a permis de décrire des cas particuliers du CECSP pouvant être résolus en temps polynomial mais aussi a permis, dans certains cas, la mise en place d'une propriété permettant de simplifier le problème en caractérisant les différentes formes que peut prendre une activité. Cette simplification du problème a ensuite été utilisée pour mettre en place des méthodes de résolution pour le CECSP, souvent adaptées de méthodes existantes dans le cadre des problèmes d'ordonnancement cumulatifs.

Les méthodes de résolution dédiées au CECSP et proposées dans ce manuscrit sont regroupées en deux catégories : les méthodes issues de la programmation par contraintes et adaptées des méthodes définies pour la contrainte cumulative et les méthodes issues de la programmation linéaire mixte et en nombres entiers adaptées des méthodes définies pour le RCPSP.

Plusieurs des techniques de programmation par contraintes utilisées dans le cadre de la résolution de la contrainte cumulative ont été décrites et, en particulier, les principaux algorithmes de filtrage qui lui sont dédiés. Une partie de ces algorithmes a été adapté au cas du CECSP. C'est le cas, par exemple, du raisonnement énergétique qui prend une part importante de ce manuscrit. Ce raisonnement comptant parmi les plus forts dans le cas de la contrainte cumulative a été le premier à avoir été adapté avec en particulier la caractérisation d'un ensemble d'intervalles suffisant pour l'application du raisonnement énergétique, de cardinalité polynomiale. De plus, les récents travaux de Derrien *et al.* permettant l'accélération de ce raisonnement ont aussi pu être adaptés. D'autres raisonnements comme le Time-Table, le raisonnement disjonctif et le raisonnement Time-Table disjonctif ont aussi pu être transformés afin de s'appliquer dans le cas du CECSP. Enfin, un nouvel algorithme de détection d'incohérence utilisant un programme linéaire basé sur un problème de flot couplé avec le raisonnement Time-

Table a été présenté. Les expérimentations conduites sur ces raisonnements – inclus à l'intérieur d'une méthode de branchement hybride – ont permis de montrer que ce nouvel algorithme de vérification peut s'avérer plus efficace en pratique que le raisonnement énergétique.

Différents modèles de programmation linéaire en nombres entiers pour le RCPSP ont été décrits dans le chapitre 4. Parmi ces modèles, le premier repose sur une formulation indexée par le temps et les deux autres sur des formulations basées sur les événements. Ces dernières ont montré leur efficacité dans le cas où l'horizon de temps des modèles devient très grand. De plus, contrairement aux formulations indexées par le temps, elles permettent de modéliser des dates de début et de fin d'activités continues. De ce fait, nous avons adapté ces modèles au cas du CECSP. Un modèle indexé par le temps est aussi détaillé. En effet, ces modèles ont prouvé leur efficacité dans le cadre du RCPSP.

Pour chacun des modèles présentés, des inégalités valides et/ou des techniques de coupes ont été présentées afin de renforcer ces derniers. Des inégalités directement déduites du raisonnement énergétique sont introduites pour les modèles indexés par le temps du CECSP et du RCPSP. Pour les modèles à événements, une comparaison des relaxations linéaires des deux modèles présentés est effectuée. Des inégalités permettant de renforcer le modèle ayant les moins bonnes relaxations, le modèle On/Off, sont présentées. Ces inégalités, appelées inégalités de non préemption, ont été montrées comme appartenant à l'enveloppe convexe du polyèdre formé par l'ensemble de toutes les affectations possibles pour les variables binaires correspondant à une seule activité. Enfin, plusieurs autres ensembles d'inégalités pour les modèles à événements ont été présentés et les performances relatives à l'ajout de différentes combinaisons de ces inégalités au modèle ont été détaillées.

Malgré tout, le CECSP reste un problème difficile, en particulier dans sa forme générale. En effet, les résultats présentés dans cette thèse nous ont permis de résoudre des instances allant jusqu'à 60 activités pour la version décisionnelle de ce problème et seulement jusqu'à 30 pour la version ayant pour objectif la minimisation de la consommation de la ressource.

Le CECSP est un nouveau problème pour lequel de nombreux travaux restent à faire. Parmi les perspectives directes des résultats présentés dans ce manuscrit, on trouve :

**La considération de fonctions de rendement plus générales.** Même si les fonctions concaves permettent une plus grande liberté d'expression du problème, elles restent insuffisantes pour modéliser certains problèmes réels. Dans un premier temps, la considération de fonctions de rendement convexes est une continuation naturelle de ce travail. Cependant, ce n'est toujours pas suffisant dans certains cas. En effet, beaucoup de fonctions de rendement ne sont ni totalement concaves, ni totalement convexes mais sont convexes sur certains intervalles et concaves sur les autres intervalles.

**La mise en place d'algorithmes de filtrage plus performants.** Qu'il s'agisse d'algorithmes plus performants en termes de temps de calcul ou plus performants en termes de filtrage, cette direction de recherche est une perspective importante. L'adaptation d'autres algorithmes de filtrage mis en place pour la contrainte cumulative semble une piste intéressante mais l'accélération du temps de calcul de ces derniers est fait au prix d'un filtrage de moins bonne qualité. Ceci peut s'avérer critique dans le cas du CECSP où les algorithmes de filtrage pour la contrainte cumulative sont beaucoup plus faibles. A l'inverse, la mise en place d'algorithmes dédiés au problème ayant un pouvoir de filtrage plus grand sera probablement effectuée au prix d'une augmentation importante du temps de calcul.

**Amélioration des modèles de programmation linéaire mixte.** Dans un premier temps, les modèles décrits dans ce manuscrit pourraient encore être renforcés. En effet, les inégalités énergétiques décrites pour le modèle indexé par le temps sont un moyen intéressant de coupler des techniques issues de la programmation par contraintes à la programmation linéaire mixte. D'autres inégalités de ce type pourraient être définies ou de meilleures techniques d'intégration de ces inégalités dans le processus de résolution pourraient être mises en place.

L'amélioration des modèles à événements est aussi une piste de recherche importante puisque, contrairement aux modèles indexés par le temps, ces derniers permettent d'obtenir des solutions exactes pour le CECSP mais au prix d'un temps de calcul beaucoup plus important. Pour renforcer ces modèles, d'autres jeux d'inégalités, plus fortes, peuvent être mises en place. Par exemple, des inégalités décrivant des facettes du polyèdre formé par l'ensemble des solutions de chaque modèle pourraient être exhibées.

**L'utilisation d'autres méthodes de résolution.** Les techniques décrites dans ce manuscrit utilisent principalement des techniques de programmation linéaire mixte et de programmation par contraintes. Cependant, d'autres paradigmes existent et des techniques issues de ces derniers pourraient être considérées. De même, des techniques appliquées sur des problèmes connexes pourraient aussi être adaptées, quitte à discrétiser le problème. Parmi ces techniques, on pourra trouver les techniques de génération de colonnes, de branch-and-cut, la programmation dynamique ou des techniques utilisant des règles de priorités, etc.

**L'étude de la version discrète du problème.** Malgré le fait que la restriction discrète du CECSP peut conduire à l'obtention de solutions non optimales, cette dernière offre de bonnes approximations dans le cas général. De ce fait, étudier la version discrète du problème peut s'avérer intéressant. De plus, les bonnes performances des méthodes discrètes, notamment du modèle indexé par le temps ou du modèle de programmation par contraintes présenté dans l'annexe A, nous poussent à diriger la recherche dans cette direction.

[ABK+13] Christian Artigues, Peter Brucker, Sigrid Knust, Oumar Koné, Pierre Lopez, and Marcel Mongeau. A note on « an event-based MILP models for resource-constrained project scheduling problems ». *Computers & Operations Research*, 40(4) :1060 – 1063, 2013.

[ADN07] Christian Artigues, Sophie Demassey, and Emmanuel Néron. *Resource-Constrained Project Scheduling : Models, Algorithms, Extensions and Applications*. ISTE, 2007.

[AL15] Christian Artigues and Pierre Lopez. Energetic reasoning for energy-constrained scheduling with a continuous resource. *J. of Scheduling*, 18(3) :225–241, June 2015.

[ALH13] Christian Artigues, Pierre Lopez, and Alain Haït. The energy scheduling problem : Industrial case-study and constraint propagation techniques. *International Journal of Production Economics*, 143(1) :13 – 23, 2013.

[ALR12] Christian Artigues, Pierre Lopez, and David Rivreau. Integer programming and constraint propagation for scheduling under energy constraints. In *27th European Conference on Operational Research (EURO 2012)*, Vilnius, 2012.

[AMR03] Christian Artigues, Philippe Michelon, and Stéphane Reusser. Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research*, 149(2) :249 – 267, 2003. Sequencing and Scheduling.

[BC02] Nicolas Beldiceanu and Mats Carlsson. A new multi-resource cumulative constraint with negative heights. In *Principles and Practice of Constraint Programming : 8th International Conference, CP 2002 Ithaca, NY, USA, September 9–13, 2002 Proceedings*, pages 63–79. Springer International Publishing, 2002.

[BD04] Philippe Baptiste and Sophie Demassey. Tight LP bounds for resource constrained project scheduling. *OR Spectrum*, 26(2) :251–262, 2004.

[BEP+01] Jacek Błażewicz, Klaus H. Ecker, Erwin Pesch, Günter Schmidt, and Jan Węglarz. *Scheduling Computer and Manufacturing Processes*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2nd edition, 2001.

[BHS11] Timo Berthold, Stefan Heinz, and Jens Schulz. *An Approximative Criterion for the Potential of Energetic Reasoning*, pages 229–239. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[BLK83] Jacek Blazewicz, Jan Karel Lenstra, and Alexander H.G. Rinnooy Kan. Scheduling subject to resource constraints : classification and complexity. *Discrete Applied Mathematics*, 5(1) :11 – 24, 1983.

[BLPN99] Philippe Baptiste, Claude Le Pape, and Wim Nuijten. Satisfiability tests and time-bound adjustments for cumulative scheduling problems. *Annals of Operations Research*, 92(0) :305–333, 1999.

[BLPN01] Philippe Baptiste, Claude Le Pape, and Wim Nuijten. *Constraint-based scheduling : applying constraint programming to scheduling problems*. International series in operations research & management science. Kluwer Academic, Boston, 2001. Réimpression : 2003.

[BO79] Jon L. Bentley and Thomas A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, 28(9) :643–647, 1979.

[Bon16] Nicolas Bonifas. A $O(n^2 \log n)$ propagation for the Energy Reasoning. In *ROADEF - 17ème congrès annuel de la Société française de recherche opérationnelle et d'aide à la*

*décision*, Compiègne, France, February 2016. Société française de recherche opérationnelle et d'aide à la décision.

[BP07] Nicolas Beldiceanu and Emmanuel Poder. A continuous multi-resource cumulative constraint with positive-negative resource consumption-production. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems : 4th International Conference, CPAIOR 2007 Brussels, Belgium, May 23-26. Proceedings*, pages 214–228. Springer Berlin Heidelberg, 2007.

[CAVT87a] Nicos Christofides, Ramó Alvarez-Valdes, and José Manuel Tamarit. Project scheduling with resource constraints : A branch and bound approach. *European Journal of Operational Research*, 29(3) :262 – 273, 1987.

[CAVT87b] Nicos Christofides, Ramón Alvarez-Valdes, and José Manuel Tamarit. Project scheduling with resource constraints : A branch and bound approach. *European Journal of Operational Research*, 29(3) :262 – 273, 1987.

[CL91] Jacques Carlier and Bruno Latapie. Une méthode arborescente pour résoudre les problèmes cumulatifs. *RAIRO - Operations Research - Recherche Opérationnelle*, 25(3) :311–340, 1991.

[DDrH00] Erik Demeulemeester, Bert De reyck, and Willy Herroelen. The discrete time/resource trade-off problem in project networks : a branch-and-bound approach. *IIE Transactions*, 32(11) :1059–1069, 2000.

[Dem03] Sophie Demassey. *Hybrid Constraint Programming-Integer Linear Programming approaches for the Resource-Constrained Project Scheduling Problem.* Theses, Université d'Avignon, December 2003.

[Der15] Alban Derrien. *Cumulative scheduling in constraint programming : energetic characterization of reasoning and robust solutions.* Theses, Ecole des Mines de Nantes, November 2015.

[DHP99] Ulrich Dorndorf, Toàn Phan Huy, and Erwin Pesch. A survey of interval capacity consistency tests for time- and resource-constrained scheduling. In *Project Scheduling : Recent Models, Algorithms and Applications*, pages 213–238. Springer US, 1999.

[DP14] Alban Derrien and Thierry Petit. *A New Characterization of Relevant Intervals for Energetic Reasoning*, pages 289–297. Springer International Publishing, Cham, 2014.

[DRDH98] Bert De Reyck, Erik Demeulemeester, and Willy Herroelen. Local search methods for the discrete time/resource trade-off problem in project networks. *Naval Research Logistics (NRL)*, 45(6) :553–578, 1998.

[FT10] Cord-Ulrich Fündeling and Norbert Trautmann. A priority-rule method for project scheduling with work-content constraints. *European Journal of Operational Research*, 203(3) :568 – 574, 2010.

[GHS15a] Steven Gay, Renaud Hartert, and Pierre Schaus. Simple and scalable time-table filtering for the cumulative constraint. In *Principles and Practice of Constraint Programming : 21st International Conference, CP 2015, Cork, Ireland, August 31 – September 4, 2015, Proceedings*, pages 149–157. Springer International Publishing, 2015.

[GHS15b] Steven Gay, Renaud Hartert, and Pierre Schaus. Time-table disjunctive reasoning for the cumulative constraint. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 157–172. Springer International Publishing, 2015.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[HE80] Robert M. Haralick and Gordon L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14(3) :263 – 313, 1980.

[JE89] Catherine Thuriot Jacques Erschler, Pierre Lopez. Scheduling under time and resource constraints. In *Proc. of Workshop on Manufacturing Scheduling, 11th IJCAI*, Detroit, USA, 1989.

[Jen06] Johan L. W. V. Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30(1) :175–193, 1906.

[KALM11] Oumar Koné, Christian Artigues, Pierre Lopez, and Marcel Mongeau. Event-based MILP Models for Resource-constrained Project Scheduling Problems. *Computers & Operations Research*, 38(1) :3–13, January 2011.

[Kis05] Tamás Kis. A branch-and-cut algorithm for scheduling of projects with variable-intensity activities. *Mathematical Programming*, 103(3) :515–539, 2005.

[Kol96] Rainer Kolisch. Serial and parallel resource-constrained project scheduling methods revisited : Theory and computation. *European Journal of Operational Research*, 90(2) :320 – 333, 1996.

[Kon09] Oumar Koné. *Nouvelles approches pour la résolution du problème d'ordonnancement de projet à moyens limités*. Theses, Université Paul Sabatier - Toulouse III, December 2009.

[KS96] Rainer Kolisch and Arno Sprecher. PSPLIB – a project scheduling problem library. *European Journal of Operational Research*, 96 :205–216, 1996.

[Lah82] Abdelkader Lahrichi. Ordonnancements. La notion de "parties obligatoires" et son application aux problèmes cumulatifs. *RAIRO - Recherche Opérationnelle*, 16 :241–262, 1982.

[LBC12] Arnaud Letort, Nicolas Beldiceanu, and Mats Carlsson. A scalable sweep algorithm for the cumulative constraint. In *Principles and Practice of Constraint Programming : 18th International Conference, CP 2012, Québec City, QC, Canada, October 8-12, 2012. Proceedings*, pages 439–454. Springer International Publishing, 2012.

[Lew08] J. Lewis. Algebra symposium : Optimizing fuel consumption. http://homepages.math.uic.edu/~jlewis/math165/asavgcost.pdf, 2008.

[MHM05] C. Le Van M.N. Hung and P. Michel. Non-convex aggregative technology and optimal economic growth. ftp://www-bsg.univ-paris1.fr/pub/mse/cahiers2005/B05095.pdf, 2005.

[MMRB98] Aristide Mingozzi, Vittorio Maniezzo, Salvatore Ricciardelli, and Lucio Bianco. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Management Sciences*, 44(5) :714–729, May 1998.

[MVH08] Luc Mercier and Pascal Van Hentenryck. Edge finding for cumulative scheduling. *INFORMS J. on Computing*, 20(1) :143–153, January 2008.

[NAKL16a] Margaux Nattaf, Christian Artigues, Tamás Kis, and Pierre Lopez. Inégalités valides pour les modèles à évènements des problèmes d'ordonnancement sous contraintes de ressource. In *ROADEF - 17ème congrès annuel de la Société française de recherche opérationnelle et d'aide à la décision*, Compiègne, France, February 2016. Société française de recherche opérationnelle et d'aide à la décision.

[NAKL16b] Margaux Nattaf, Christian Artigues, Tamás Kis, and Pierre Lopez. Polyhedral results and valid inequalities for resource-constrained scheduling problem event-based models. In *29th Conference of the European Chapter on Combinatorial Optimization*, Budapest, Hungary, May 2016.

[NAL15a] Margaux Nattaf, Christian Artigues, and Pierre Lopez. Flow and energy based satisfiability tests for the continuous energy-constrained scheduling problem with concave piecewise linear functions. In *CP Doctoral Program 2015*, pages 70–81, Cork, Ireland, September 2015.

[NAL15b] Margaux Nattaf, Christian Artigues, and Pierre Lopez. A hybrid exact method for a scheduling problem with a continuous resource and energy constraints. *Constraints*, 20(3) :304–324, 2015.

[NAL+15c] Margaux Nattaf, Christian Artigues, Pierre Lopez, Rosa Medina, Lorena Pradenas, and Victor Parada. A batch sizing and scheduling problem on parallel machines with different speeds, maintenance operations, setup times and energy costs. In *International Conference on Industrial Engineering and Systems Management (IESM 2015)*, Seville, Spain, October 2015.

[NAL16] Margaux Nattaf, Christian Artigues, and Pierre Lopez. Programmation linéaire mixte et programmation par contraintes pour un problème d'ordonnancement à contraintes énergétiques. In *Douzièmes Journées Francophones de Programmation par Contraintes (JFPC)*, Montpellier, France, 2016.

[NALR16] Margaux Nattaf, Christian Artigues, Pierre Lopez, and David Rivreau. Energetic reasoning and mixed-integer linear programming for scheduling with a continuous resource and linear efficiency functions. *OR Spectrum*, 38(2) :pp. 459–492, March 2016.

[NK14] Anulark Naber and Rainer Kolisch. MIP models for resource-constrained project scheduling with flexible resource profiles. *European Journal of Operational Research*, 239(2) :335 – 348, 2014.

[NKAL16] Margaux Nattaf, Tamás Kis, Christian Artigues, and Pierre Lopez. Polyhedral results and valid inequalities for the Continuous Energy-Constrained Scheduling Problem. Research report, laas - cnrs, November 2016.

[Nui94] Wim Nuijten. *Time and resource constrained scheduling : A constraint satisfaction approach.* PhD thesis, Technische Universiteit Eindhoven, 1994.

[OG93] Ramón Alvarez-Valdés Olaguíbel and José Manuel Tamarit Goerlich. The project scheduling polyhedron : Dimension, facets and lifting theorems. *European Journal of Operational Research*, 67(2) :204 – 220, 1993.

[OQ13] Pierre Ouellet and Claude-Guy Quimper. Time-table extended-edge-finding for the cumulative constraint. In *Principles and Practice of Constraint Programming : 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, pages 562–577. Springer International Publishing, 2013.

[PV10] Vincent Van Peteghem and Mario Vanhoucke. A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, 201(2) :409 – 418, 2010.

[PWW69] A. Alan B. Pritsker, Lawrence J. Waiters, and Philip M. Wolfe. Multiproject scheduling with limited resources : A zero-one programming approach. *Management Science*, 16(1) :93–108, 1969.

[RK07] Mohammad R. Ranjbar and Fereydoon Kianfar. Solving the discrete time/resource trade-off problem in project scheduling with genetic algorithms. *Applied Mathematics and Computation*, 191(2) :451 – 456, 2007.

[RRK09] Mohammad Ranjbar, Bert De Reyck, and Fereydoon Kianfar. A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling. *European Journal of Operational Research*, 193(1) :35 – 48, 2009.

[Tes16] Alexander Tesch. *A Nearly Exact Propagation Algorithm for Energetic Reasoning in* $O(n^2 \log n)$, pages 493–519. Springer International Publishing, Cham, 2016.

[Vil09a] Petr Vilím. Edge finding filtering algorithm for discrete cumulative resources in $O(kn \log n)$. In *Principles and Practice of Constraint Programming : 15th International Conference, CP 2009 Lisbon, Portugal, September 20-24, 2009 Proceedings*, pages 802–816. Springer International Publishing, 2009.

[Vil09b] Petr Vilím. Max energy filtering algorithm for discrete cumulative resources. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems : 6th International Conference, CPAIOR 2009 Pittsburgh, PA, USA, May 27-31, 2009. Proceedings*, pages 294–308. Springer International Publishing, 2009.

[Vil11] Petr Vilím. Timetable edge finding filtering algorithm for discrete cumulative resources. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems : 8th International Conference, CPAIOR 2011, Berlin, Germany, May 23-27, 2011. Proceedings*, pages 230–245. Springer International Publishing, 2011.

[Wal11] Grzegorz Waligóra. Heuristic approaches to discrete-continuous project scheduling problems to minimize the makespan. *Computational Optimization and Applications*, 48(2) :399–421, 2011.

[Weg80] Jan Weglarz. Multiprocessor scheduling with memory allocation - a deterministic approach. *IEEE Transactions on Computers*, C-29(8) :703–709, Aug 1980.

[Węg81] Jan Węglarz. Project scheduling with continuously-divisible, doubly constrained resources. *Management Science*, 27(9) :1040–1053, 1981.

[Wol98] Laurence A Wolsey. *Integer programming*. Wiley, 1998.

# Annexes

*Cette partie, contenant les annexes de ce manuscrit, présente des travaux réalisés pendant la thèse mais pas assez aboutis ou trop éloignés du sujet de ce manuscrit pour y figurer à part entière. Cette partie se divise en deux sous-parties. La première correspond à l'annexe A et présente l'étude du cas discret du problème d'ordonnancement continu à contraintes énergétiques (article publié aux Journées Francophones de Programmation par Contraintes [NAL16]). Ce paragraphe présente notamment un modèle de programmation par contraintes pour la restriction discrète du problème.*
*La seconde annexe (ann. B) porte sur la mise en place d'une matheuristique pour un problème industriel d'ordonnancement avec contraintes et objectifs énergétiques. Ce travail s'inscrit dans le cadre d'une collaboration pour le projet franco-chilien ECOS C13E04 et a été publié dans la conférence IESM (International Conference on Industrial Engineering and Systems Management [NAL+15c]). Ce paragraphe présente tout d'abord le problème étudié, puis, dans un second temps, trois modèles de programmation linéaire à événements sont introduits. Les deux premiers, constituant des méthodes de résolution exacte, comportent un nombre de variables et de contraintes trop important pour espérer les résoudre en temps raisonnable. De ce fait, un troisième modèle simplifié est présenté et la solution de ce modèle est utilisée comme solution de base des modèles exacts.*

# Annexe A

# Programmation linéaire mixte et programmation par contraintes pour un problème d'ordonnancement à contraintes énergétiques

Nous considérerons un problème d'ordonnancement cumulatif dans lequel les tâches ont une durée et un profil de consommation de ressource variable. Ce profil, qui peut varier en fonction du temps, est une variable de décision du problème dont dépend la durée de la tâche associée. Pour ce problème NP-difficile, nous présentons un modèle de programmation par contraintes et un modèle de programmation linéaire en nombres entiers (PLNE). De plus, des inégalités valides déduites de la programmation par contraintes viennent renforcer le PLNE. Ces modèles sont ensuite comparés par le biais d'expérimentations.

## A.1 Introduction

Nous étudions un problème d'ordonnancement avec ressource continue et contraintes énergétiques, le Continuous Energy-Constrained Scheduling Problem (CECSP). Dans ce problème, un ensemble de tâches $\mathcal{A} = \{1, \ldots, n\}$ utilisant une ressource continue et cumulative de capacité limitée $B$ doit être ordonnancé. La quantité de ressource nécessaire à l'exécution d'une tâche n'est pas fixée mais - le profil de consommation de cette dernière est une fonction $b_i(t)$ définie pour tout $t \in \mathbb{R}$ [1] - doit être déterminé. Une fois la tâche commencée et jusqu'à sa date de fin, la fonction $b_i(t)$ doit être comprise entre une valeur maximale, $r_i^{max}$, et minimale, $r_i^{min}$.

De plus, la consommation, à un instant $t$, d'une partie de la ressource permet la production d'une certaine quantité d'énergie et, une tâche finit lorqu'elle a reçu une énergie $W_i$. Cette énergie est calculée par le biais d'une fonction de rendement $f_i$, propre à chaque tâche. Dans cet article, ces fonctions sont supposées continues, croissantes, affines et peuvent être exprimées de la manière suivante :

$$f_i(b) = \begin{cases} 0 & \text{si } b = 0 \\ a_i * b + c_i & \text{si } r_i^{min} = 0 \text{ et } b \in ]r_i^{min}, r_i^{max}] \\ a_i * b + c_i & \text{si } r_i^{min} \neq 0 \text{ et } b \in [r_i^{min}, r_i^{max}] \end{cases}$$

avec $a_i > 0$ et $c_i \geq -a_i * r_i^{min}$ pour s'assurer que $f_i(b) \geq 0,\ \forall b \in [r_i^{min}, r_i^{max}[$.

Dans la suite, nous dénotons par $est_i$ et $lst_i$ la date de début au plus tôt et au plus tard de $i$ et

1. Le domaine de définition de la fonction peut être réduit mais, pour faciliter les notations, nous supposons qu'elle est définie pour tout $t \in \mathbb{R}$.

par $eet_i$ et $let_i$ la date de fin au plus tôt et au plus tard de $i$.

Pour trouver une solution pour le CECSP, nous devons déterminer, pour chaque tâche $i \in \mathcal{A}$, sa date de début $st_i$, sa date de fin $et_i$ et sa fonction d'allocation de ressource $b_i(t)$, $\forall t \in \mathcal{T} = [\min_{i\in\mathcal{A}} est_i, \max_{i\in\mathcal{A}} let_i]$. De plus, ces variables doivent satisfaire les contraintes suivantes :

$$est_i \leq st_i < et_i \leq let_i \qquad \forall i \in \mathcal{A} \tag{A.1}$$

$$r_i^{min} \leq b_i(t) \leq r_i^{max} \qquad \forall i \in \mathcal{A},\ \forall t \in [st_i, et_i] \tag{A.2}$$

$$b_i(t) = 0 \qquad \forall i \in \mathcal{A},\ \forall t \notin [st_i, et_i] \tag{A.3}$$

$$\int_{st_i}^{et_i} f_i(b_i(t))dt = W_i \qquad \forall i \in \mathcal{A} \tag{A.4}$$

$$\sum_{i\in\mathcal{A}} b_i(t) \leq B \qquad \forall t \in \mathcal{T} \tag{A.5}$$

L'objectif auquel nous nous sommes intéressés est la minimisation de la consommation totale de la ressource. Dans [NALR16], les auteurs montrent que trouver une solution admissible pour le CECSP est déjà un problème NP-complet.

De plus, une instance ayant des données seulement entières peut n'avoir que des solutions à valeurs dans $\mathbb{R}$ [NALR16]. Cependant, une dilatation de l'instance, i.e. multiplier les données par un certain coefficient $\alpha$, permet de palier à ce problème. De ce fait et dans le but de résoudre des instances entières, nous nous sommes intéressés, dans un premier temps, à la version discrète du CECSP, le DECSP (Discrete Energy Constrained Scheduling Problem). Dans ce problème, toutes les données sont supposées entières et les domaines de chaque variable ne contiennent que des valeurs entières, i.e. $st_i,\ et_i,\ b_i(t) \in \mathbb{N}$ et $b_i(t)$ est défini $\forall t \in \mathcal{T}_\mathcal{D} = \{\min_{i\in\mathcal{A}} est_i, \ldots, \max_{i\in\mathcal{A}} let_i\}$.

Pour ce problème, nous présentons un modèle de programmation par contraintes (PPC) permettant l'utilisation des algorithmes de propagation mis en place pour la contrainte cumulative, notamment [GHS15]. Un modèle de programmation linéaire en nombres entiers (PLNE) est aussi présenté. Ce modèle est ensuite renforcé à l'aide d'inégalités valides déduites du raisonnement énergétique [EL90]. Ces deux modèles sont ensuite testés sur des instances à données entières, avec et sans dilatation.

## A.2 Modèle de programmation par contraintes

Pour modéliser le DECSP à l'aide de la PPC, nous divisons chaque tâche $i$ en deux sous-tâches $i_{min}$ et $i_{preem}$. La première, $i_{min}$, est une tâche ayant une consommation de ressource fixe, égale à $r_i^{min}$, et une durée variable $p_i$. Cette tâche représente la quantité de ressource obligatoirement consommée par une activité durant son exécution, i.e. $r_i^{min}$. La seconde, $i_{preem}$ est une tâche préemptive optionnelle, consommant une quantité variable de ressource comprise entre 0 et $r_i^{max} - r_i^{min}$ et devant s'exécuter en même temps que $i_{min}$. Cette tâche est elle-même divisée en sous-tâches $i^\ell_{preem},\ \ell \in \{1, \ldots, et_i - st_i\} = \mathcal{L}_i$. Notons que $|\mathcal{L}_i| = et_i - st_i \leq \lceil \frac{W_i}{f_i(r_i^{min})} \rceil$.

**Exemple A.2.1.** *Considérons la tâche possédant les attributs suivant : $est_i = 0$, $let_i = 6$, $W_i = 28$, $r^{min} = 1$, $r^{max} = 5$ et $f_i(b) = 2b + 1$. La Figure A.1 présente un ordonnancement de cette tâche (à droite) et l'ordonnancement correspondant donné par le modèle (à gauche) avec $r_{i^2_{preem}} = 0$.*

Le problème du DECSP peut alors être formulé à l'aide des variables :

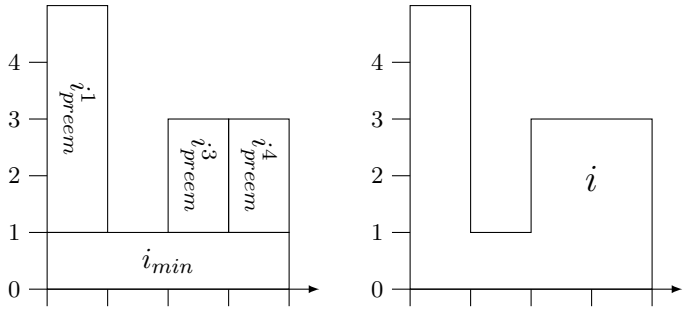



FIGURE A.1 – Exemple de solution du modèle PPC.

— $i_{min} = \{s_{i_{min}}, e_{i_{min}}, b_{i_{min}}, p_{i_{min}}\},\ \forall i \in \mathcal{A}$
— $i^{\ell}_{preem} = \{s_{i^{\ell}_{preem}}, e_{i^{\ell}_{preem}}, b_{i^{\ell}_{preem}}, p_{i^{\ell}_{preem}}\},\ \forall i \in \mathcal{A},\ \ell \in \mathcal{L}_i.$

et des contraintes :

1. $\forall (i,l) \in \mathcal{A} \times \mathcal{L}_i :\ e_{i^{\ell}_{preem}} = s_{i^{\ell+1}_{preem}}$
2. $\forall i \in \mathcal{A} :\ s_{i_{min}} = s_{i^1_{preem}}$ et $e_{i^{|\mathcal{L}_i|}_{preem}} = e_{i_{min}}$
3. $\forall t \in \mathcal{T} :$
$$\sum_{\substack{i \in \mathcal{A} \\ t \in [s_{i_{min}}, e_{i_{min}}[}} b_{i_{min}} + \sum_{\substack{(i,l) \in \mathcal{A} \times \mathcal{L}_i \\ t \in [s_{i^{\ell}_{preem}}, e_{i^{\ell}_{preem}}[}} b_{i^{\ell}_{preem}} \leq B$$
4. $\forall i \in \mathcal{A} :\ \sum_{l \in \mathcal{L}_i} \left( f_i(b_{i^{\ell}_{preem}})(e_{i^{\ell}_{preem}} - s_{i^{\ell}_{preem}}) \right) + f_i(b_{i_{min}})(e_{i_{min}} - s_{i_{min}}) \geq W_i$

La première contrainte permet d'ordonner les sous-tâches de $i_{preem}$. Ceci dans le but de faciliter la modélisation des autres contraintes. La seconde contrainte modélise le fait que $i_{preem}$ commence et finit en même temps que $i_{min}$. La troisième contrainte assure que la capacité de la ressource n'est pas excédée en sommant, à un instant $t$, les consommations minimales des tâches en cours ainsi que les consommations des sous-tâches préemptives en cours. Enfin, la quatrième contrainte permet de s'assurer que chaque tâche reçoit au moins l'énergie requise $W_i$.

Un des avantages de cette formulation est qu'elle permet l'utilisation des algorithmes de propagation mis en places pour la contrainte cumulative tels que le time-table classique [BLPN01], disjonctif [GHS15] ou associé au edge-finding [Vil11], le raisonnement disjonctif [BLPN01], ou encore le raisonnement énergétique [EL90]. Cependant, certains de ces raisonnements peuvent être adaptés pour prendre en compte l'ensemble du problème. C'est le cas, par exemple, du raisonnement énergétique détaillé ci-dessous.

### A.2.1 Raisonnement énergétique

Ce paragraphe présente un algorithme de propagation pour le DECSP basé sur le raisonnement énergétique défini pour le CECSP [NALR16]. L'adaptation de ce raisonnement au cas discret est quasi-directe. Cependant, nous rappelons les bases de celui-ci car nous l'utiliserons dans la suite pour déduire des inégalités valides pour le PLNE.

Le principe du raisonnement énergétique est de comparer la quantité de ressource disponible dans un intervalle avec la quantité minimale de ressource consommée par toutes les tâches dans cet intervalle.

Les configurations pour lesquelles la quantité de ressource requise par une tâche $i$ dans l'intervalle $[t_1, t_2[$ est minimale correspondent toujours à une configuration où la tâche reçoit le maximum d'énergie possible, i.e. est ordonnancée à $r_i^{max}$, en dehors de $[t_1, t_2[$, tout en respectant les contraintes (A.1)–(A.5). Ceci correspond donc à une des configurations suivantes :

— la tâche est calée à gauche : ordonnancée à $r_i^{max}$ durant $[est_i, t_1[$ ;
— la tâche est calée à droite : ordonnancée à $r_i^{max}$ durant $[t_2, let_i[$ ;
— la tâche est centrée : ordonnancée à $r_i^{max}$ durant $[est_i, t_1[\cup[t_2, let_i[$ ou ordonnancée à $r_i^{min}$ durant $[t_1, t_2[$.

En effet, dans le dernier cas, il peut arriver qu'ordonnancer la tâche à $r_i^{max}$ dans $[est_i, t_1[\cup[t_2, let_i[$ implique que la quantité d'énergie restant à apporter à la tâche dans $[t_1, t_2[$ ne soit pas suffisante pour ordonnancer la tâche à $r_i^{min}$ durant $[t_1, t_2[$. Or, ceci impliquerait une violation de la contrainte (A.2). Dans ce cas, la tâche est donc ordonnancée à $r_i^{min}$ durant l'intervalle $[t_1, t_2[$. Alors la quantité de ressource requise par la tâche $i$ dans $[t_1, t_2[$ est la quantité minimale requise par ces configurations.

Les intervalles $[t_1, t_2[$ sur lesquels appliquer ce test pour le CECSP sont décrits dans [NALR16]. Pour le DECSP, nous devons considérer les projections de ces intervalles sur les entiers, i.e. $[a, b[ \rightarrow [\lfloor a \rfloor, \lceil b \rceil[$. Les ajustements pour le CECSP s'adaptent aussi naturellement au DECSP à l'aide de cette même projection.

## A.3 Modèle de programmation linéaire en nombres entiers

### A.3.1 Modèle

La formulation proposée dans cet article est une formulation indexée par le temps. Elle est adaptée de la formulation décrite dans [NALR16]. Dans ces formulations, l'horizon de temps est divisé en intervalles de taille 1 et est défini par : $\mathcal{T}_\mathcal{D}$. Pour chaque activité $i \in \mathcal{A}$ et pour chaque instant $t \in \mathcal{T}_\mathcal{D}$, nous définissons deux variables binaires $x_{it}$ et $y_{it}$ pour modéliser le début et la fin des activités. La variable $x_{it}$ (resp. $y_{it}$) prendra la valeur 1 si et seulement si l'activité $i$ commence (finit) à l'instant $t$. Pour modéliser la consommation de ressource et l'apport en énergie, nous introduisons deux variables, $b_{it}$ et $w_{it}$ qui représentent respectivement la quantité de ressource consommée par l'activité $i$ dans la période de temps $t$ et l'énergie reçue par cette même activité durant cette période.

Par manque de place, ce modèle n'est pas entièrement décrit ici mais nous décrivons les contraintes permettant de lier les variables $b_{it}$ et $w_{it}$, i.e. permettant de calculer l'énergie apportée à $i$ dans la période $t$, $w_{it}$ , en fonction de la consommation de ressource $b_{it}$. Nous donnons aussi le nombre de variables et de contraintes du modèle.

Les contraintes liant $b_{it}$ et $w_{it}$, $\quad \forall t \in \mathcal{T}_\mathcal{D},\ \forall i \in \mathcal{A}$ sont les suivantes :

$$w_{it} = a_i b_{it} + c_i \left( \sum_{\tau=est_i}^{t} x_{i\tau} - \sum_{\tau=est_i+1}^{t} y_{i\tau} \right) \tag{A.6}$$

Cette contrainte nous permet de modéliser la fonction de rendement $f_i$, $\forall i \in \mathcal{A}$. En effet, $\left(\sum_{\tau=est_i}^{t} x_{i\tau} - \sum_{\tau=est_i+1}^{t} y_{i\tau}\right)$ est égale à 1 si et seulement si l'activité $i$ est en cours à l'instant $t$. Dans ce cas là, la valeur de l'énergie apportée à $i$ est bien $w_{it} = a_i b_{it} + c_i$. Le second cas se produit quand l'activité $i$

n'est pas en cours à $t$. Dans ce cas, $b_{it} = 0$ implique $w_{it} = 0$.

Le modèle possède donc $2n|\mathcal{T}_\mathcal{D}|$ variables binaires, $2n|\mathcal{T}_\mathcal{D}|$ variables continues et au plus $3n + |\mathcal{T}_\mathcal{D}| * (6n + 1)$ contraintes.

## A.4 Inégalités valides basées sur le raisonnement énergétique

Ce paragraphe décrit des inégalités valides déduites du raisonnement énergétique pour le PLNE. Soit $\mathcal{R}$ l'ensemble des intervalles d'intérêt pour le raisonnement énergétique.

$$(x_{iest_i} + y_{ilet_i} - 1)\,\underline{b}(i,t_1,t_2) + \sum_{j\neq i}\underline{b}(j,t_1,t_2) \leq$$
$$B(t_2 - t_1) \quad \forall i \in \mathcal{A},\ \forall [t_1,t_2] \in \mathcal{R} \tag{A.7}$$

$$(x_{iest_i} + \sum_{t=t_1}^{t_2} y_{it} - 1)\,\underline{b}(i,t_1,t_2) + \sum_{j\neq i}\underline{b}(j,t_1,t_2) \leq$$
$$B(t_2 - t_1) \quad \forall i \in \mathcal{A},\ \forall [t_1,t_2] \in \mathcal{R} \tag{A.8}$$

$$(\sum_{t=t_1}^{t_2} x_{it} + y_{ilet_i} - 1)\,\underline{b}(i,t_1,t_2) + \sum_{j\neq i}\underline{b}(j,t_1,t_2) \leq$$
$$B(t_2 - t_1) \quad \forall i \in \mathcal{A},\ \forall [t_1,t_2] \in \mathcal{R} \tag{A.9}$$

$$(1 - \sum_{t<t_1} x_{it} - \sum_{t>t_2} y_{it})\,\underline{b}(i,t_1,t_2) + \sum_{j\neq i}\underline{b}(j,t_1,t_2) \leq$$
$$B(t_2 - t_1) \quad \forall i \in \mathcal{A},\ \forall [t_1,t_2] \in \mathcal{R} \tag{A.10}$$

$$(\sum_{t\leq t_1} x_{it} + \sum_{t\geq t_2} y_{it} - 1)\,\underline{b}(i,t_1,t_2) \leq B(t_2 - t_1)$$
$$\forall i \in \mathcal{A},\ \forall [t_1,t_2] \in \mathcal{R} \tag{A.11}$$

L'inégalité (A.7) correspond au cas où la tâche est centrée et est ordonnancée à $r_i^{max}$ durant $[est_i, t_1] \cup [t_2, let_i[$. En effet, cette inégalité n'est active que dans le cas où $(x_{iest_i} + y_{ilet_i} - 1) = 1 \Rightarrow [x_{iest_i} = 1 \wedge y_{ilet_i} = 1]$. Or, ceci implique que la tâche commence à $est_i$ et finit $let_i$. Donc, la ressource disponible dans $[t_1, t_2[$ doit être suffisante pour donner la quantité de ressource minimale requise par $i$ dans $[t_1, t_2[$ dans cette configuration. Dans tous les autres cas, l'inégalité devient $\sum_{j\neq i}\underline{b}(j,t_1,t_2) \leq B(t_2 - t_1)$ ou $\sum_{j\neq i}\underline{b}(j,t_1,t_2) - \underline{b}(i,t_1,t_2) \leq B(t_2 - t_1)$.

Les inégalités (A.8), (A.9), (A.10), (A.11) correspondent respectivement au cas où $i$ est calée à gauche, $i$ est calée à droite, $i$ est complètement incluse dans $[t_1, t_2]$, $i$ est exécutée à $r_i^{min}$ durant l'intervalle $[t_1, t_2]$ et sont déduites de la même façon que (A.7). Ces inégalités seront ajoutées au modèle indexé par le temps décrit à la section A.3 pour renforcer ce dernier.

## A.5 Résultats expérimentaux

Nous avons testé les différentes méthodes de résolution proposées dans cet article sur les instances de [NALR16]. Les expérimentations ont été conduites sous le système d'exploitation Ubuntu 64-bit 12.04 et les résultats sont calculés au moyen d'un processeur 4-core, 8 thread Core (TM) i7-4770 CPU

| | | $1^{ère}$ sol. | | fin algo. | |
|---|---|---|---|---|---|
| | #tâches | temps(s) | écart | temps | %opt. |
| DEF | 20 | 5.37 | 7.85 | 75.4 | 0.25 |
| ER | 20 | 8.4 | 10.6 | 78.9 | 0.22 |
| DEF | 25 | 4.6 | 4.4 | 83.8 | 0.17 |
| ER | 25 | 0.06 | 3.86 | 60.1 | 0.4 |
| DEF | 30 | 0.99 | 7.18 | 75.19 | 0.25 |
| ER | 30 | 5.66 | 7.53 | 75.8 | 0.25 |

TABLE A.1 – Résultats du PLNE avec et sans inégalités valides : ER et DEF resp. (TL 1000s)

et de 8GB de mémoire RAM.

Le modèle de PLNE est résolu à l'aide de IBM Cplex 12.6 avec 2 threads et une limite de temps de 100 secondes. Les inégalités déduites du raisonnement énergétique sont calculées avant la résolution du PLNE et ajoutées statiquement au modèle. Ceci augmente la taille du modèle de $5|\mathcal{R}|n$ contraintes (avec $|\mathcal{R}| \in O(n^2)$).

Le tableau B.1 décrit les résultats du PLNE.

L'ajout des inégalités du raisonnement énergétique permet de résoudre les instances à 25 tâches de manières plus efficace. Cependant, elles ralentissent le modèle pour les instances à 20 ou 30 tâches mais la perte de rapidité dans ce cas là est beaucoup moins élevée que le gain fait sur les instances à 25. Une poursuite de recherche intéressante serait d'essayer d'ajouter ces contraintes pendant la résolution du PLNE en tant que coupes.

Le modèle de PPC est résolu avec IBM CP Optimizer 12.6. Le tableau B.2 décrit les résultats du modèle de PPC. Le modèle de PPC est testé sans ajout du raisonnement énergétique présentés dans cet article mais le modèle utilise les propagateurs du solveur. Des résultats expérimentaux plus détaillées seront proposés lors de la conférence.

Les résultats montrent l'intérêt des inégalités valides ajoutées au PLNE. Le modèle de PPC ne permet pas de prouver l'optimalité des solutions trouvées mais a des résultats similaires au PLNE. En effet, dans presque tous les cas, le modèle de PPC trouve une solution aussi bonne que le PLNE, sans toutefois prouver son optimalité.

Les méthodes présentées ont aussi été testées sur des instances dilatées dans le but de garantir l'existence de solutions entières. La dilatation est effectuée de la manière suivante. Soit $\alpha$ le plus petit commun multiple à tous les $r_i^{min}$ et $r_i^{max}$. Alors, la dilatation consiste à multiplier $let_i$, $eet_i$, $lst_i$, $est_i$ et $W_i$ par $\alpha$. Ces expérimentations n'ont pas donné de résultats dû à la grande taille de ces modèles. Les modèles continus pourraient donc rester la seule alternative pour obtenir des solutions optimales

| | $1^{ère}$ sol. | | fin algo. | |
|---|---|---|---|---|
| #tasks | time | deviation | time lim. | %solved |
| 20 | 0.19 | 34.1 | 100 | 95 |
| 25 | 0.3 | 47.9 | 100 | 91 |
| 30 | 0.42 | 43.1 | 100 | 95 |

TABLE A.2 – Résultats du modèle PPC

dans le cas où la solution est réelle.

Parmi les poursuites de recherche possibles, on trouve l'amélioration des modèles avec la réduction du nombre de variable et/ou de contraintes et la mise en place d'algorithmes de propagation dédiés.

## Bibliographie

[BLPN01] Philippe Baptiste, Claude Le Pape, and Wim Nuijten. *Constraint-based scheduling : applying constraint programming to scheduling problems.* International series in operations research & management science. Kluwer Academic, Boston, 2001. Réimpression : 2003.

[EL90] J. Erschler and P. Lopez. Energy-based approach for task scheduling under time and resources constraints. In *2nd International Workshop on Project Management and Scheduling*, pages 115–121, Compiègne, France, 1990.

[GHS15] Steven Gay, Renaud Hartert, and Pierre Schaus. Time-table disjunctive reasoning for the cumulative constraint. In *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*, pages 157–172. Springer International Publishing, 2015.

[NALR16] Margaux Nattaf, Christian Artigues, Pierre Lopez, and David Rivreau. Energetic reasoning and mixed-integer linear programming for scheduling with a continuous resource and linear efficiency functions. *OR Spectrum*, 38(2) :pp. 459–492, March 2016.

[Vil11] Petr Vilím. Timetable edge finding filtering algorithm for discrete cumulative resources. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems : 8th International Conference, CPAIOR 2011, Berlin, Germany, May 23-27, 2011. Proceedings*, pages 230–245. Springer International Publishing, 2011.

## Annexe B

# A batch sizing and scheduling problem on parallel machines with different speeds, maintenance operations, setup times and energy costs

This paper considers a production scheduling problem in a Chilean company from the metalworking industry. This company produces steel balls of different diameters on parallel production lines. There are different types of production lines and each production line may have a different speed for producing each diameter. Furthermore a setup time occurs when changing the diameter produced on each machine. Besides these production and setup operations, maintenance operations have to be scheduled. These electrical machines yield high energy demands. It is therefore crucial to minimize total energy consumption, which depends on batch/machine assignment, and maximum demand on peak hours. We consider the batch sizing and scheduling problem involving electricity costs in a non-uniform parallel machine context. Given a demand for each family of steel balls, the problem consists in splitting the demand in sublots (batches) that have to be assigned and scheduled on the parallel machines together with the required maintenance operations. The goal is to complete the schedule before a common deadline while minimizing electricity costs. We propose to tackle this problem through mixed integer linear programming. We propose a global formulation and a two-phase matheuristic. Computational results on realistic instances are provided.

### B.1 Introduction

This paper considers a production scheduling problem in a Chilean company from the metalworking industry. The problem and the industrial context was described in [Urr14]. This company produces steel balls on parallel production lines. The steel balls are obtained by roll forming or forging from a raw material consisting of metal bars. The steel balls are mainly used in the copper and gold mining industry for mineral grinding, i.e. for reducing the size of the mineral particles to a maximal granularity that permits to remove the major part of impurities from the mineral. There are several types of steel balls to produce, each corresponding to a different ball diameter. There are two types of production lines : roll formers for small diameters and forges for larger diameters (although medium diameter balls can be produced by both production line types). Each production line may have a different speed for producing each diameter. Furthermore a setup time occurs when changing the diameter produced on each machine. Besides these production and setup operations, maintenance operations have to be scheduled. These machines are electrical and the production process results in high energy

demands. As mentioned in [Urr14], up to 50% of the production cost in such a manufacturing process can be due to electricity consumption. In Chile, as in many other places, different electricity rates are applied for peak hours and off-peak hours. So it can be crucial for metalworking companies to control the electricity demand during peak hours. Planning the maintenance and setup operations during the peak hours can be intuitively a policy that favors electricity cost decrease.

In this paper we consider a batch sizing and scheduling problem involving electricity costs in a non-uniform parallel machine context. Given a demand for each family of steel balls, the problem consists in splitting the demand in sublots (batches) that have to be assigned and scheduled on the parallel machines together with the required maintenance operations. The goal is to complete the schedule before a common deadline while minimizing electricity costs. We propose to tackle this problem through a mixed integer linear programming approach.

Section B.2 presents the batch sizing and scheduling problem. Section B.3 is devoted to a brief presentation of the related work. Section B.4 gives the proposed mixed-integer linear programming (MILP) formulation. Section B.5 gives a simplified MILP formulation that ignores the peak costs and that serves as a basis for a matheuristic. Section B.6 presents the considered realistic problem instances and the results obtained by our approach. Section B.7 draws concluding remarks and directions for future work.

## B.2 The industrial scheduling problem

The problem involves a set $\mathcal{J}$ of $n$ lots (jobs) to be scheduled on a set $\mathcal{M}$ of $m$ machines. There is total a production demand $D_j$ for each job $j \in \mathcal{J}$ to be fulfilled during the scheduling horizon given by a time interval $[0, T]$. Each job $j \in \mathcal{J}$ can be split into a maximum number of $\lfloor D_j/\epsilon \rfloor$ sublots (batches), where $\epsilon$ is the minimum batch size. The machines able to produce a job $j$ are gathered in set $\mathcal{M}_j$. For a given job $j \in \mathcal{J}$, a machine $k \in \mathcal{M}_j$ has a production speed $v_j^k$. Whenever a batch of a job $j \in \mathcal{J}$ is scheduled immediately after a batch of job $i \in \mathcal{J}$ on a machine $k \in \mathcal{M}$, a setup time $s_{ij}^k \geq 0$ is necessary on the machine. Each machine is assumed to have a constant electrical power demand $w_k$. There are $H$ peak periods within the horizon, each being defined by an interval $[a_h, f_h)$ with $a_h < f_h$ for $h = 1, \ldots, H$ and $f_h < a_{h+1}$ for $h = 1, \ldots, H-1$. Concerning cost minimization, we considered a weighted sum problem where $\alpha$ denotes the weight of the total energy consumption while $\beta$ denotes the weight of the maximum consumption during peak hours.

A solution consists in determining :

— A set of batches $\mathcal{B}_j$ with $|\mathcal{B}_j| = B_j \geq 1$, for each job $j \in \mathcal{J}$,

— A production amount $q_{j,b} \geq \epsilon$, for each batch $b \in \mathcal{B}_j$,

— The assignment $a_{j,b}$ of each batch $b \in \mathcal{B}_j$, i.e. the line on which batch $b \in \mathcal{B}_j$ is produced,

— A start time $S_{j,b}$, for each job $j \in \mathcal{J}$ and for each batch $b \in \mathcal{B}_j$.

Let $\mathcal{B} = \cup_{j \in \mathcal{J}} \{(j, b) | b \in \mathcal{B}_j\}$ denote the set of pairs (job, batch index). Let $\mathcal{A}_k = \{(j, b) | a_{j,b} = k\}$ the set of batches assigned to machine $k \in \mathcal{M}$.

The problem constraints can be stated as follows :

A batch can only be assigned on an authorized machine.

$$a_{j,b} \in \mathcal{M}_j \quad \forall j \in \mathcal{J}, \forall b \in \mathcal{B}_j \tag{B.1}$$

The demand must be satisfied for each job.

$$\sum_{b \in \mathcal{B}_j} q_{j,b} = D_j \quad \forall j \in \mathcal{J} \tag{B.2}$$

The duration $p_{j,b}$ of a batch $b \in \mathcal{B}_j$ is equal to its production divided by the speed of its assigned machine.

$$p_{j,b} = q_{j,b} / v_j^{a_{j,b}} \quad \forall j \in \mathcal{J}, \forall b \in \mathcal{B}_j \tag{B.3}$$

No batches assigned to the same machine may overlap and the setup time between two consecutive batches on the same machine must be respected.

$$\begin{aligned} CrS_{j,b} \geq S_{i,b'} + p_{i,b'} + s_{i,j}^{a_{j,b}} \\ \forall (i,j) \in \mathcal{J}^2, \forall (b,b') \in \mathcal{B}_i \times \mathcal{B}_j \text{ such that } a_{i,b'} = a_{j,b} \end{aligned} \tag{B.4}$$

All production must fit in the time horizon.

$$S_{j,b} + p_{j,b} \leq T \quad \forall (j,b) \in \mathcal{B} \tag{B.5}$$

We need not introduce maintenance operations in the model. We consider that there is a subset $\mathcal{J}^M$ of the jobs $\mathcal{J}$ that correspond to maintenance operations. They concern a single machine ($|\mathcal{M}_j| = 1$). The maintenance duration is given by $D_j$ and we have $v_j^k = 1$ for $k \in \mathcal{M}_j$. We denote by $\mathcal{B}^M$ the set of batches corresponding to maintenance operations. We introduce last the constraints that each maintenance job cannot be splitted :

$$|\mathcal{B}_j| = 1 \quad \forall j \in \mathcal{J}^M \tag{B.6}$$

We assume that there is no required setup time between a maintenance job and a production job. However me make the realistic assumption that the maintenance duration is such that $D_j \geq s_{ii'}^k$, $\forall j \in \mathcal{J}^M$, $\forall i, i' \in \mathcal{J} \setminus \mathcal{J}^M, i \neq i', \forall k \in \mathcal{M}$, i.e. is larger than any other setup time. Furthermore we assume that all other setup times satisfy the triangle inequality

$$s_{ij}^k \geq s_{il}^k + s_{lj}^k \quad \forall k \in \mathcal{M}, \forall i,j,l \in \mathcal{J} \setminus \mathcal{J}^M, i \neq j \neq l$$

A solution is feasible if and only if it satisfies constraints B.1–B.6.

There are two components linked to energy consumption. As the machines have different speeds and different power consumptions, the first part of the objective is to minimize the total electricity

consumption $C$.

$$C = \sum_{k \in \mathcal{M}} w_k \sum_{(j,b) \in \mathcal{A}_k \setminus \mathcal{B}^M} p_{j,b} \tag{B.7}$$

We are also interested in the maximum demand over the peak periods. Let us define for convenience the set of batches in process at a given time point $t$

$$\mathcal{B}(t) = \{(j,b) \in \mathcal{B} | S_j, b \leq t < S_{j,b} + p_{j,b}\} \tag{B.8}$$

Then the maximum demand over peak hours is equal to

$$P_{\max} = \max_{h \in \mathcal{H}} \max_{t \in [a_h, b_h)} \sum_{k \in \mathcal{M}, \exists (j,b) \in \mathcal{B}(t) \cap \mathcal{A}_k \setminus \mathcal{B}^M} w_k \tag{B.9}$$

Remark that maintenance operations are excluded in the computation of these costs. Then the problem objective can be written

$$\min \alpha C + \beta P_{\max} \tag{B.10}$$

Note that the problem is NP-hard as the problem of finding a solution has the decision variant of the parallel machine problem as special case. The above-defined constraints can be used to check the feasibility and cost of a solution in polynomial time but they cannot be implemented as is in a mathematical programming solver.

We present a small example and its optimal solution. Consider $n = 3$ jobs on $m = 2$ lines with demands $D_1 = 10$, $D_2 = 12$, $D_3 = 6$. There is one peak period $h = 1$ with $a_h = 2$ and $b_h = 4$ and a common deadline of $T = 6$ for all jobs and maintenance operations. There are two maintenance operations $\mathcal{J}^M = \{4, 5\}$ of duration $D_4 = 2$ on machine 1 and $D_5 = 1$ on machine 2 to be performed also before the deadline. Machine power demands are $w_1 = w_2 = 10$. For each job, authorized machines are $\mathcal{M}_1 = \{1, 2\}$, $\mathcal{M}_2 = \{1\}$, $\mathcal{M}_3 = \{2\}$, $\mathcal{M}_3 = \{2\}$, $\mathcal{M}_4 = \{1\}$, $\mathcal{M}_5 = \{2\}$. Setup and speed matrices are respectively equal to

$$(s^1_{i,j})_{i,j \in \mathcal{J}} = (s^2_{i,j})_{i,j \in \mathcal{J}} = \begin{bmatrix} 0 & 2 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

$$(v^k_j)_{j \in \mathcal{J}, k \in \mathcal{M}} = \begin{bmatrix} 2 & 3 \\ 6 & 0 \\ 0 & 2 \end{bmatrix}$$

Figure B.1 presents an optimal solution for this example where $\mathcal{B}_1 = \{1, 2\}$ (job 1 is split into two batches), $\mathcal{B}_2 = \mathcal{B}_3 = 1$ (jobs 2 and 3 are scheduled in one batch). Batch machine assignments are given by $a_{1,1} = M_1$, $a_{1,2} = M_2$, $a_{2,1} = M_1$ and $a_{3,1} = M_2$. Productions inside each batch are given by $q_{1,1} = 4$, $q_{1,2} = 6$, $q_{2,1} = 12$, $q_{3,1} = 6$. According to the assigned machine speeds, the batch duration are given by $p_{1,1} = q_{1,1}/v^1_1 = 2$, $p_{1,2} = q_{1,2}/v^2_1 = 2$, $p_{2,1} = q_{2,1}/v^1_2 = 2$, $p_{3,1} = q_{3,1}/v^2_3 = 3$. The start times of the different batches and of the maintenance operations (in gray) are displayed in the Gantt chart. According to the machine power demand, total consumption of machine 1 is $4 \times w_1 = 40$ and

total consumption of machine 2 is $5 \times 10 = 50$. Hence we obtain a total energy consumption of 90 and a maximal demand on peak hours of 10. We observe that batch splitting has been necessary to meet the deadline and minimize energy consumption, while maintenance has been used to avoid large demands during peak hours.

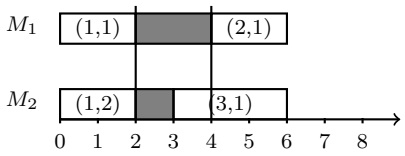



FIGURE B.1 – Optimal solution of a simple instance.

Hence, after a brief presentation of the related work, we will introduce a mixed-linear integer programming formulation.

## B.3 Related work and modeling issues

A vast literature considers upper level lot-sizing and scheduling problems in parallel machine environment with setup times (such as in [JAL11, XZZG13]) involving, among others, inventory costs and also using discrete time mixed-integer linear programming formulations. First, the discrete time models do not fit our problem as the duration of the sublot is a function of the sublot size (a continuous variable) and the machine speed for the family of balls corresponding to the sublot. Furthermore we do not have holding costs in our model. More generally, our problem is in fact related to the (operational) production scheduling level where it is assumed that lots of products aiming at satisfying the demand in a given period have already been constituted at the (tactical) planning level.

Lot streaming (also called lot splitting) is a known technique to improve the performance of production scheduling environment at the operational level [DPL97, CC05] by allowing to split the jobs across the different production stages or lines. In our case, the lot streaming process in simplified by the fact that we have a single stage. This is also the reason why we rather use the "batch sizing" term rather than lot streaming or splitting (we borrowed this term from [HKS14]).

The possibility of splitting the jobs (lots/batches) adds another dimension to the decisions besides assignment of the jobs to the machine and sequencing/scheduling of the job operations. Many lot streaming/batch sizing and scheduling problems have been considered in the literature for different scheduling context, including parallel machine scheduling for which efficient algorithms are available [Ser96, XZ00]. However, these first studies did not consider the presence of setup times that can be needed on a machine between two sublots. This feature, that occurs in our case when the ball diameter is changed, puts obviously a limit to the interest of systematic job splitting. Several approaches can be found for parallel machine lot streaming/batch sizing problems with setup times [YC03, TYCA06, BGGG08]. Compared to these previous works, our study is concerned with electricity cost minimization objectives. In manufacturing, energy constraints and costs are becoming crucial. Consequently, energy-related objectives have been recently considered both in lot-sizing problems [ADPKS+13] and in production scheduling problems [MYT07, ALH13, GDL15]. However to our knowledge, reducing the energy consumption cost by an integrated management of lot streaming/batch sizing, setup and maintenance scheduling in a parallel machine scheduling environment is

a novel approach.

In this paper, one of the objective is to come up with a mixed-integer linear programming formulation (MILP) of the problem. In scheduling problems, there are standardly three categories of MILP formulations [QS94] : the continuous time formulations with sequencing variables, the continuous time formulations with positional (or event-based) variables and the discrete time formulations. In the considered problem, the continuous lot splitting possibility and the variable machine speed combined with large time horizons would render a discrete time formulation (such as the ones generally used in the lot-sizing literature) impractical. Therefore we need a continuous time formulation. Now looking at the conceptual continuous time formulation (B.1–B.10), most non-linear constraints could be linearized in a standard way yielding a continuous time (also called disjunctive) formulation with sequencing variables $x_{j,b,i,b'} \in \{0,1\}$ where $x_{j,b,i,b'} = 1$ if and only if batch $(i,b')$ is scheduled after batch $(j,b)$ on the same machine. In [RON01], the authors consider such a formulation for a simple identical parallel machine scheduling problem with job-splitting and sequence-dependent setup times with the makespan criterion. In this simple case splitting a job is only useful if all sublots are scheduled on different machines, hence the authors consider binary sequencing variables of the type $x_{i,j,k}$ where $x_{i,j,k} = 1$ is the sublot of job $i$ in machine $k$ is sequenced before the sublot of job $j$ in machine $k$. However in our case, it can be necessary to have two batches of the same job on the same machine, especially due to the interest of avoiding production during the peak hours. Among many other configurations Fig. B.2 shows an optimal solution for a 1 machine and 3 jobs example ($m = 1$ and $n = 3$) with setup times $s_{12} = s_{21} = s_{13} = s_{31} = 1$, $s_{23} = s_{32} = 2$, unit machine speed, unit machine power, demands $D_1 = 2$, $D_2 = D_3 = 1$, deadline $T = 8$ and weights $\alpha = 0$, $\beta = 1$. Setup times are shown in gray. The optimal solution (of cost 0) needs to split job 1 in two batches to have no production within the peak hour (interval marked with a P).

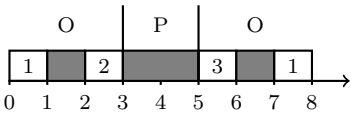



FIGURE B.2 – Two batches of a job on the same machine.

Another issue for modeling the problem in continuous time is the linearization of the expressions (B.8, B.9) defining the maximum demand over the continuous horizon. An important remark is that the electricity consumption only changes at a batch start or end time. This clearly brings us to consider an event-based model. These models consider a finite number of events and use an event-indexed continuous variable to associate a start time to each event and a binary variable to assign a batch start or end time. Consequently, we can also associate a continuous variable representing the total electricity consumption at a given event, that shall not change until the next event. Note that event-based models were previously used for lot streaming and scheduling in hybrid flow-shops [DC12] and are also widely used in batch scheduling in the process industry [FL05]. They were also more recently introduced in resource-constrained project scheduling problems [KALM13] and in scheduling problems under energy constraints [NAL15].

## B.4 A continuous time event-based mixed-integer linear programming formulation

The event-based model we propose for the problem, is based on the fact that energy consumption can only change at the beginning or at the end of each batch. Hence we fix for each job its maximum number of batches $B_j$ and we consider a set $\mathcal{E}$ of $2\sum_{j=1}^{n} B_j$ events. Now the start-end event-based formulation of the problem involves the following decision variables (variables with domain $[0,1]$ are continuous variables that will be set binary by the constraints involving other binary variables).

— $t_e \geq 0$ is the time of event $e \in \mathcal{E}$.
— $x_{j,b}^e \in \{0,1\}$ is equal to 1 iff batch $(j,b) \in \mathcal{B}$ starts at event $e \in \mathcal{E}$.
— $y_{j,b}^e \in \{0,1\}$ is equal to 1 iff batch $(j,b) \in \mathcal{B}$ ends at event $e \in \mathcal{E}$.
— $a_{j,b}^k \in \{0,1\}$ is equal to 1 iff batch $(j,b) \in \mathcal{B}$ is assigned to machine $k \in \mathcal{M}$.
— $x_{j,b}^{e,k} \in [0,1]$ auxiliary variable equal to product $x_{j,b}^e a_{j,b}^k$.
— $y_{j,b}^{e,k} \in [0,1]$ auxiliary variable equal to product $y_{j,b}^e a_{j,b}^k$.
— $o_{e,k} \in [0,1]$ is equal to 1 iff machine $k \in \mathcal{M}$ is on at event $e \in \mathcal{E}$.
— $p_{j,b} \geq 0$ is the production of batch $(j,b) \in \mathcal{B}$.
— $P_{e,h} \in \{0,1\}$ models the relative positioning of intervals $[t_e, t_{e+1})$ and peak hour interval $[a_h, f_h)$ in the sense that $P_{e,h} = 0 \Rightarrow t_e \geq f_h$.
— $P_{h,e} \in \{0,1\}$ models the relative positioning of intervals $[t_e, t_{e+1})$ and peak hour interval $[a_h, f_h)$ in the sense that $P_{h,e} = 0 \Rightarrow t_{e+1} \leq a_h$.
— $P_e^k \in [0,1]$ models the fact that event $e$ is active on a peak hour on machine $k$ (i.e. $[t_e, t_{e+1})$ has a non-empty intersection with a peak hour interval and machine $k$ is on at event $e$).
— $W_{j,b} \geq 0$ total electrical consumption of batch $(j,b)$.

The constraints can be expressed as follows. $M$ denotes a sufficiently large integer.

$$t_e \leq t_{e+1} \quad \forall e \in \mathcal{E} \setminus \{|\mathcal{E}|\} \tag{B.11}$$

Each batch $(j,b)$ has to be scheduled at most once.

$$\sum_{e \in \mathcal{E}} x_{j,b}^e \leq 1 \quad \forall (j,b) \in \mathcal{B} \tag{B.12}$$

$$\sum_{e \in \mathcal{E}} y_{j,b}^e \leq 1 \quad \forall (j,b) \in \mathcal{B} \tag{B.13}$$

A started batch must be ended, and reciprocally.

$$\sum_{e \in \mathcal{E}} y_{j,b}^e = \sum_{e \in \mathcal{E}} x_{j,b}^e \quad \forall (j,b) \in \mathcal{B} \tag{B.14}$$

The end event of a batch cannot be lower than its start event.

$$\sum_{f \in \mathcal{E}, f \geq e} x_{j,b}^f \geq \sum_{f \in \mathcal{E}, f \geq e} y_{j,b}^f \quad \forall (j,b) \in \mathcal{B}, \forall e \in \mathcal{E} \tag{B.15}$$

A batch has to be assigned to a machine to have a non-zero production (as $B_j$ is the maximum number of batch for job $j$, there can be some unassigned batches).

$$p_{j,b} \leq \sum_{k \in \mathcal{M}_j} a_{j,b}^k D_j \quad \forall (j,b) \in \mathcal{B} \tag{B.16}$$

A batch assigned to a machine must have a minimum size.

$$p_{j,b} \geq \sum_{k \in \mathcal{M}_j} a_{j,b}^k \epsilon \quad \forall (j,b) \in \mathcal{B} \tag{B.17}$$

A batch has to be assigned to at most one machine among the candidate machines.

$$\sum_{k \in \mathcal{M}_j} a_{j,b}^k \leq 1 \quad \forall (j,b) \in \mathcal{B} \tag{B.18}$$

The following constraints are set on auxiliary variables $x_{j,b}^{e,k}$ and $y_{j,b}^{e,k}$ to linearize products $x_{i,b}^e a_{j,b}^k$ and $y_{i,b}^e a_{j,b}^k$.

$$x_{j,b}^{e,k} \geq x_{j,b}^e + a_{j,b}^k - 1 \quad \forall (j,b) \in \mathcal{B}, \forall e \in \mathcal{E}, \forall k \in \mathcal{M} \tag{B.19}$$

$$x_{j,b}^{e,k} \leq x_{j,b}^e \quad \forall (j,b) \in \mathcal{B}, \forall e \in \mathcal{E}, \forall k \in \mathcal{M} \tag{B.20}$$

$$x_{j,b}^{e,k} \leq a_{j,b}^k \quad \forall (j,b) \in \mathcal{B}, \forall e \in \mathcal{E}, \forall k \in \mathcal{M} \tag{B.21}$$

$$y_{j,b}^{e,k} \geq y_{j,b}^e + a_{j,b}^k - 1 \quad \forall (j,b) \in \mathcal{B}, \forall e \in \mathcal{E}, \forall k \in \mathcal{M} \tag{B.22}$$

$$y_{j,b}^{e,k} \leq x_{j,b}^e \quad \forall (j,b) \in \mathcal{B}, \forall e \in \mathcal{E}, \forall k \in \mathcal{M} \tag{B.23}$$

$$y_{j,b}^{e,k} \leq a_{j,b}^k \quad \forall (j,b) \in \mathcal{B}, \forall e \in \mathcal{E}, \forall k \in \mathcal{M} \tag{B.24}$$

A machine is on at an event $e$ depending on its status at the preceding event and/or start or completion of a task on this machine (we define also $o_{0,k} = \sum_{(j,b) \in \mathcal{B}} x_{j,b}^{0,k}$, $\forall k \in \mathcal{M}$).

$$\begin{aligned} Cro_{e,k} = o_{e-1,k} &- \sum_{(j,b) \in \mathcal{B}} y_{j,b}^{e,k} \\ &+ \sum_{(j,b) \in \mathcal{B}} x_{j,b}^{e,k} \quad \forall e \in \mathcal{E}, \forall k \in \mathcal{M} \end{aligned} \tag{B.25}$$

The common deadline must be satisfied.

$$t_{|\mathcal{E}|} \leq T \tag{B.26}$$

Setup times must be satisfied between two consecutive batches.

$$t_f \geq t_e + s^k_{ij}(x^{f,k}_{j,b} + y^{e,k}_{i,b'} - 1) \quad \forall e, f \in \mathcal{E}, f \geq e, \\ \forall (j,b),(i,b') \in \mathcal{B}, (j,b) \neq (i;b'), \forall k \in \mathcal{M} \tag{B.27}$$

Batches can be assigned only on authorized machines.

$$a^k_{j,b} = 0 \quad \forall (j,b) \in \mathcal{B}, \forall k \in \mathcal{M} \setminus \mathcal{M}_j \tag{B.28}$$

End and start time events of a batch have to be spaced according to the batch duration on the assigned machine.

$$t_f \geq t_e + p_{j,b}/v_{j,k} - M(2 - x^{e,k}_{j,b} - y^{f,k}_{j,b}) \\ \forall e, f \in \mathcal{E}, f \geq e, \forall (j,b) \in \mathcal{B}, \forall k \in \mathcal{M} \tag{B.29}$$

$$t_f \leq t_e + p_{j,b}/v_{j,k} + M(2 - x^{e,k}_{j,b} - y^{f,k}_{j,b}) \\ \forall e, f \in \mathcal{E}, f \geq e, \forall (j,b) \in \mathcal{B}, \forall k \in \mathcal{M} \tag{B.30}$$

The demand of each job must be satisfied.

$$\sum_{b \in \mathcal{B}_j} p_{j,b} = D_j \quad \forall j \in \mathcal{J} \tag{B.31}$$

The following constraints set the relative positioning of event $e$ and peak hour period $h$, in the sense that if $P_{e,h} = 0$, intersection of $[t_e, t_e + 1)$ and peak hour interval $[a_h, f_h)$ is empty because $t_e \geq f_h$

$$t_e \geq f_h(1 - P_{e,h}) \quad \forall e \in \mathcal{E}, \forall h \in \mathcal{H} \tag{B.32}$$

The following constraints set the relative positioning of event $e$ and peak hour period $h$, in the sense that if $P_{h,e} = 0$, intersection of $[t_e, t_e + 1)$ and peak hour interval $[a_h, f_h)$ is empty because $t_{e+1} \leq a_h$

$$t_{e+1} - a_h \leq T P_{h,e} \quad \forall e \in \mathcal{E}, \forall h \in \mathcal{H} \tag{B.33}$$

Machine $k$ is marked as active during peak hour at event $e$ whenever machine $k$ is on at event $e$ and interval $[t_e, t_{e+1})$ intersects peak hour interval $[a_h, f_h)$.

$$P^k_e \geq P_{e,h} + P_{h,e} + o_{e,k} - 2 \\ \forall e \in \mathcal{E}, \forall h \in \mathcal{H}, \forall k \in \mathcal{M} \tag{B.34}$$

The maximum demand objective is defined by the following constraints.

$$P_{\max} \geq \sum_{k \in \mathcal{M}} w_k P^k_e \quad \forall e \in \mathcal{E} \tag{B.35}$$

The consumption of a batch depends on its duration and on the power of its assigned machine.

$$W_{j,b} \geq w_k p_{j,b}/v_{j,k} - M(1 - a^k_{j,b}) \quad \forall (j,b) \in \mathcal{B} \setminus \mathcal{B}^M, \forall k \in \mathcal{M} \tag{B.36}$$

The objective function aims at minimizing the total consumption and the maximum demand on peak hours.

$$\min \alpha \sum_{(j,b)\in\mathcal{B}\setminus\mathcal{B}^M} W_{j,b} + \beta P_{\max} \tag{B.37}$$

The MILP model made of constraints (B.11-B.37) is denoted as (MILP1). We now present an alternative model that contains more binary variables but less constraints. Product variables $x^{e,k}_{j,b}$ and $y^{e,k}_{j,b}$ can be considered as binary and assignment variables $a^k_{j,b}$, as well as start and end variables $x^e_{j,b}$ and $y^e_{j,b}$ can be eliminated by setting :

$$a^k_{j,b} = \sum_{e\in\mathcal{E}} x^{e,k}_{j,b} \quad \forall (j,b) \in \mathcal{B}, \forall k \in \mathcal{M}$$

$$x^e_{j,b} = \sum_{k\in\mathcal{M}} x^{e,k}_{j,b} \quad \forall (j,b) \in \mathcal{B}, \forall e \in \mathcal{E}$$

$$y^e_{j,b} = \sum_{k\in\mathcal{M}} y^{e,k}_{j,b} \quad \forall (j,b) \in \mathcal{B}, \forall e \in \mathcal{E}$$

The obtained model is denoted by (MILP2).

If we come back to Figure B.1 example. We can use 5 global events with $t_1 = 0$, $t_2 = 2$, $t_3 = 3$, $t_4 = 4$ and $t_5 = 6$. For MILP2, we have $x^{1,1}_{1,1} = y^{2,1}_{1,1} = 1$, $x^{1,2}_{1,2} = y^{2,2}_{1,2} = 1$, $x^{4,1}_{2,1} = y^{5,1}_{2,1} = 1$, $x^{3,2}_{3,1} = y^{5,2}_{3,1} = 1$ and for maintenance operations $x^{2,1}_{4,1} = y^{4,1}_{4,1} = 1$ and $x^{3,2}_{5,1} = y^{5,2}_{5,1} = 1$. For peak computation the only concerned event is $e = 3$ and we have $P^2_3 = 1$ because $P_{3,1} + P_{1,3} + o_{3,2} = 1$ (the event is inside the peak and the machine is on).

## B.5 A simplified model and a matheuristic

The models (MILP1) and (MILP2) proposed in Section B.4 have several drawbacks. First they have a large number of binary variables and constraints, especially constraints involving big-$M$ coefficient that significantly weaken the LP relaxation. Second, finding a solution that ends before the common deadline can be difficult and even impossible in practice.

If we ignore the maximum consumption during peak hours, we can simplify the model, transforming the event-based model into a positional model. We consider a set $\mathcal{E}_k$ of maximum $\sum_{j=1}^n B_j$ positions on each machine $k$ and the following reduced set of variables, including a tardiness variable for each job aiming at relaxing the hard common deadline and including it into the objective function.

— $t_{e,k} \geq 0$ is the start time of batch scheduled at position $e \in \mathcal{E}_k$ on machine $k \in \mathcal{M}$.
— $x^{e,k}_{j,b} \in \{0,1\}$ is equal to 1 if batch $(j,b) \in \mathcal{B}$ is assigned to position $e$ on machine $k$
— $p^{e,k}_{j,b} \geq 0$ is the production of batch $(j,b) \in \mathcal{B}$ if it is scheduled at position $e$ on machine $k$.

— $T_k \geq 0$ is the tardiness of machine $k \in \mathcal{M}$ in the case it finishes its production after the common deadline.

The problem constraints can then be expressed as follows. The start times of tasks at each position are ordered.

$$t_{e,k} \leq t_{e+1,k} \quad \forall k \in \mathcal{M}, \forall e \in \mathcal{E}_k \setminus \{|\mathcal{E}_k|\} \tag{B.38}$$

Each batch $(j, b)$ has to be scheduled at most once.

$$\sum_{k \in \mathcal{M}_j} \sum_{e \in \mathcal{E}_k} x_{j,b}^{e,k} \leq 1 \quad \forall (j,b) \in \mathcal{B} \tag{B.39}$$

Each event on each machine $(e, k)$ corresponds to at most one batch.

$$\sum_{(j,b) \in \mathcal{B}} x_{j,b}^{e,k} \leq 1 \quad \forall k \in \mathcal{M}_j, \ \forall e \in \mathcal{E}_k \tag{B.40}$$

The production of a batch $(j, b)$ at a position of a machine is 0 if the batch is not assigned at this position on this machine.

$$p_{j,b}^{e,k} \leq D_j x_{j,b}^{e,k} \quad \forall (j,b) \in \mathcal{B}, \forall k \in \mathcal{M}_j, \forall e \in \mathcal{E}_k \tag{B.41}$$

A batch assigned to a position on a machine must have a minimum size of $\epsilon$.

$$p_{j,b}^{e,k} \geq \epsilon x_{j,b}^{e,k} \quad \forall (j,b) \in \mathcal{B}, \forall k \in \mathcal{M}_j, \forall e \in \mathcal{E}_k \tag{B.42}$$

The demand of each job must be satisfied.

$$\sum_{b \in \mathcal{B}_j} \sum_{k \in \mathcal{M}_j} \sum_{e \in \mathcal{E}_k} p_{j,b}^{e,k} = D_j \quad \forall j \in \mathcal{J} \tag{B.43}$$

Setup times must be satisfied between two consecutive batches.

$$t_{e+1,k} \geq t_{e,k} + \sum_{(j,b) \in \mathcal{B}} p_{i,b}^{e,k} / v_{i,k} + s_{ij}^k (\sum_{b=1}^{B_i} x_{i,b}^{e,k} + \sum_{b'=1}^{B_j} x_{j,b'}^{e+1,k} - 1)$$
$$\forall e \in \mathcal{E}_k \setminus |\mathcal{E}_k|, \forall i, j \in \mathcal{J}, i \neq j, \forall k \in \mathcal{M} \tag{B.44}$$

The following constraint computes the violation of the deadline (tardiness) on machine k (in case a job is assigned to the last event, its duration must be added).

$$T_k \geq t_{|\mathcal{E}_k|} + \sum_{(j,b) \in \mathcal{B}} p_{j,b}^{e,k} - T \quad \forall k \in \mathcal{M} \tag{B.45}$$

The objective function aims at minimizing a weighted sum of the total tardiness and the total consumption.

$$\min \gamma \sum_{k \in \mathcal{M}} T_k + \sum_{(j,b) \in \mathcal{B}} \sum_{k \in \mathcal{M}_j} \sum_{e \in \mathcal{E}_k} w_k p_{j,b}^{e,k} / v_{j,k} \tag{B.46}$$

where $\gamma$ is a coefficient large enough to ensure that the total tardiness will be minimized in priority.

This model, denoted as (MILP3) is greatly simplified, however the solution ignores peak hours. Consequently in a second phase, The solution of (MILP3) can serve as a basis of a matheuristic for further improvement of the computed peak cost. To that purpose we compute a global event set $\mathcal{E}' = \cup_{k\in\mathcal{M}}\mathcal{E}_k$ and we sort the events according to their time $t_e$. Then we can accelerate the solution time of (MILP1) or (MILP2) by preassigning the start and end times of activities to the event set according to the solution found by (MILP3).

## B.6 Computational experiments

The experiments are conducted on an Intel Core i7-4770 processor with 4 cores and 8 gigabytes of RAM under the 64-bit Ubuntu 12.04 operating system. We use CPLEX 12.6 with 1 thread for solving the models. The total time limit of 3050 seconds for solving both (MILP2) and (MILP3) models : 3000 seconds for (MILP3) and 50 seconds for (MILP2).

The instances are extracted from the real instances of [Urr14] and are as follows. For these instances, the number of machines is equal to 3, the time horizon is equal to the number of hours in each month (e.g. in January, this number is $31 \times 24 = 744$) and the machine consumption is constant and equal to $10MW$ for each machine.

Each instance has 10 different lots and the setup time and the speed of each machine are described by Table B.1 and Table B.2 respectively. In Table B.2, whenever the speed of the machine is equal to zero then the lots can not be scheduled on this machine.

| i \ j | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 4 | 5 | 6 | 7 | 8 | 1 | 1 | 0 | 1 |
| 2 | 4 | 0 | 4 | 5 | 6 | 7 | 1 | 1 | 0 | 1 |
| 3 | 5 | 4 | 0 | 4 | 5 | 6 | 1 | 1 | 1 | 1 |
| 4 | 6 | 5 | 4 | 0 | 4 | 5 | 0 | 2 | 2 | 2 |
| 5 | 7 | 6 | 5 | 4 | 0 | 4 | 1 | 3 | 3 | 3 |
| 6 | 8 | 7 | 6 | 5 | 4 | 0 | 1 | 4 | 3 | 4 |
| 7 | 4 | 4 | 5 | 4 | 4 | 5 | 0 | 5 | 2 | 5 |
| 8 | 4 | 4 | 5 | 6 | 7 | 8 | 5 | 0 | 2 | 5 |
| 9 | 4 | 4 | 5 | 6 | 7 | 8 | 5 | 3 | 0 | 3 |
| 10 | 4 | 4 | 5 | 6 | 7 | 8 | 5 | 5 | 3 | 0 |

TABLE B.1 – Setup times between batches.

For each instance, the maintenance duration is equal to 120 for each machine. As in Chile, peak hours occurs from 6pm to 11pm, we set the peak hours to these values in our instances.

Unfortunately, both model (MILP1) and (MILP2) can not be used alone to find an optimal solution. Indeed, the great number of constraints and variables prevent the solver from finding a solution, even a feasible one. Therefore, in order to solve the problem, we use first (MILP3) and then, we use the solution to help (MILP2) to find a feasible solution.

At the beginning of the procedure, i.e. before using (MILP3), we use a simple heuristic to help the

| lot \ line | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 0 | 4 | 0 |
| 2 | 0 | 5.1 | 0 |
| 3 | 0 | 6 | 0 |
| 4 | 8.1 | 8.9 | 0 |
| 5 | 9.5 | 9 | 0 |
| 6 | 11 | 8.8 | 0 |
| 7 | 10.6 | 0 | 4 |
| 8 | 0 | 0 | 5.4 |
| 9 | 0 | 0 | 8.5 |
| 10 | 0 | 0 | 8.1 |

TABLE B.2 – Machine speeds for each lot.

model finding a solution. This heuristic considers only one batch for each different lot and schedule this batch on the least loaded machine. This heuristic is described by algorithm 4. Maintenance operations are randomly set in such a way that, on each machine, the end of the operation occurs before the deadline, i.e. the end of the planning horizon $T$.

**Algorithme 4 :** Simple heuristic for finding a simple solution.

$machLoad_k = 0, \ \forall k \in \mathcal{M}$
$aff_j = -1, \ \forall j \in \mathcal{J}$
**for** *every lot* $j$ **do**
  $minLoad \leftarrow \infty$
  $LeastLoadedMach \leftarrow -1$
  **for** *all machine* $k$ **do**
    **if** $k \in \mathcal{M}_j$ *and* $machLoad_k < minLoad$ **then**
      $minLoad \leftarrow machLoad_k$
      $LeastLoadedMach \leftarrow k$
  $aff_j = LeastLoadedMach$
  $machLoaded_{aff_j} = machLoaded_{aff_j} + D_j/v_j^k$

After solving (MILP3), we use the solution found as a basis solution of (MILP2). To make sure this solution is feasible, we use a relaxation of (MILP2). Indeed, the solution of (MILP3) may violate the deadline (constraints (B.26)). Therefore, we remove this constraint from the model and add the tardiness in the objective. Then, the objective function of (MILP2) becomes :

$$\min \alpha \sum_{(j,b)\in\mathcal{B}\setminus\mathcal{B}^M} W_{j,b} + \beta P_{\max} + \gamma T_{ard} \tag{B.47}$$

where $T_{ard} \geq T - t_{|E|}$ and $T_{ard} \geq 0$. Coefficients $\alpha$, $\beta$ and $\gamma$ are set in such a way that priority is given to the minimization of the tardiness and then to the minimization of the maximum peak hour consumption and of the total consumption equivalently. Note that this relaxation can be seen as a bi-objective optimization problem.

Experiments have been done on 10 instances and for different numbers of batches, i.e. the number

| | MILP3 | | | | MILP2 | | | |
|---|---|---|---|---|---|---|---|---|
| #batch | consump. | tard. | gap | time(s) | peak | tard. | consump. | gap |
| 2 | 16711,54 | 19,39 | 49,56 | 25,2 | 30 | 18,41 | 16692,6 | 59,88 |
| 3 | 16703,83 | 21,43 | 49,71 | 25,49 | 30 | 22,44 | 16699,8 | 71,21 |
| 4 | 16743,19 | 27,41 | 56,8 | 29,22 | 30 | 26,85 | 16750,71 | 97,18 |

TABLE B.3 – Results of experiments for (MILP2+MILP3).

of sublots for each lot : 2, 3 and 4.

Results are displayed in Table B.3. The first column displays the number of batches. The second and seventh column show the total consumption found by (MILP2) and (MILP3) respectively. The third and eighth column present the tardiness of the solution returned by (MILP2) and (MILP3) respectively. The fourth and ninth column display the gap of the solution found by (MILP2) and (MILP3) respectively. The fifth column presents the time needed to solve (MILP2). For (MILP3), since none of the instances are solved optimally, this time is equal to 3000s and is not displayed in the table. For each number of batches, the first model solved 50% of the instances optimally. Finally, the maximum peak of consumption at the end of the algorithm (after (MILP3)) is shown in column six.

As we can see, the model solves all the instances but not optimally. (MILP2) solves half of the instances to optimal and (MILP3) reduces either the total consumption or the tardiness for 2 and 3 batches. Another remark can be made about the results of the experiments. Indeed, the peak of consumption is very high. This is mostly due to the large size of the second model.

Table B.4 [1] presents the detailed results of experiments on eight instances. The column are as in Table B.3.

## B.7 Conclusion

In this paper, we consider a real industrial problem. For this problem, we present three different models. Both of them solve exactly the problem (MILP1 and MILP2) whereas the last one only minimizes the total consumption (MILP3). To solve the problem, we use first the (MILP3) and we use the solution to help the (MILP2) to improve the maximum consumption during peak hours.

The third model is quite efficient as it solved most of the instances very quickly but the second model does not improve the maximum consumption within the time limit allowed. Therefore, an important amount of work is left to be done. There are multiple research direction to pursue. Indeed, an efficient solution method minimizing the maximum consumption during peak hours needs to be developed. For this purpose, several heuristics can be tested in order to improve the quality of the solution in the beginning of the resolution or between the two MILP models.

Another development can be made by improving the modeling of the problem with a Mixed Integer Linear Program in order to decrease the number of constraints and/or variables to simplify the solution procedure. Finally, we aim at designing extended formulations and to develop column generation procedures.

1. in Appendix.

## Acknowledgment

This work was funded by ECOS–CONICYT French-Chile cooperation program C13E04.

## Bibliography

[ADPKS+13] Nabil Absi, Stéphane Dauzère-Pérès, Safia Kedad-Sidhoum, Bernard Penz, and Christophe Rapine. Lot sizing with carbon emission constraints. *European Journal of Operational Research*, 227(1) :55–61, 2013.

[ALH13] Christian Artigues, Pierre Lopez, and Alain Haït. The energy scheduling problem : Industrial case-study and constraint propagation techniques. *International Journal of Production Economics*, 143(1) :13–23, 2013.

[BGGG08] Patrizia Beraldi, Gianpaolo Ghiani, Antonio Grieco, and Emanuela Guerriero. Rolling-horizon and fix-and-relax heuristics for the parallel machine lot-sizing and scheduling problem with sequence-dependent set-up costs. *Computers & Operations Research*, 35(11) :3644–3656, 2008.

[CC05] Jen Huei Chang* and Huan Neng Chiu. A comprehensive review of lot streaming. *International Journal of Production Research*, 43(8) :1515–1536, 2005.

[DC12] Fantahun Melaku Defersha and Mingyuan Chen. Mathematical model and parallel genetic algorithm for hybrid flexible flowshop lot streaming problem. *The International Journal of Advanced Manufacturing Technology*, 62(1-4) :249–265, 2012.

[DPL97] Stephane Dauzere-Peres and Jean-Bernard Lasserre. Lot streaming in job-shop scheduling. *Operations Research*, 45(4) :584–595, 1997.

[FL05] Christodoulos A Floudas and Xiaoxia Lin. Mixed integer linear programming in process scheduling : Modeling, algorithms, and applications. *Annals of Operations Research*, 139(1) :131–162, 2005.

[GDL15] Grigori German, Chloé Desdouits, and Claude Le Pape. Energy optimization in a manufacturing plant. In *ROADEF 2015*, Marseille, 2015.

[HKS14] Oncu Hazir and Safia Kedad-Sidhoum. Batch sizing and just-in-time scheduling with common due date. *Annals of Operations Research*, 213(1) :187–202, 2014.

[JAL11] Ross JW James and Bernardo Almada-Lobo. Single and parallel machine capacitated lotsizing and scheduling : New iterative mip-based neighborhood search heuristics. *Computers & Operations Research*, 38(12) :1816–1825, 2011.

[KALM13] Oumar Koné, Christian Artigues, Pierre Lopez, and Marcel Mongeau. Comparison of mixed integer linear programming models for the resource-constrained project scheduling problem with consumption and production of resources. *Flexible Systems and Management Journal*, 25(1-2) :25–47, 2013.

[MYT07] Gilles Mouzon, Mehmet B Yildirim, and Janet Twomey. Operational methods for minimization of energy consumption of manufacturing equipment. *International Journal of Production Research*, 45(18-19) :4247–4271, 2007.

[NAL15] M. Nattaf, C. Artigues, and P. Lopez. A hybrid exact method for a scheduling problem with a continuous resource and energy constraints. *Constraints*, DOI : 10.1007/s10601-015-9192-z, 2015.

[QS94] M. Queyranne and A.S. Schulz. Polyhedral approaches to machine scheduling. Technical Report 409/1994, Technical University of Berlin, 1994.

[RON01] G. Riotteau, O.Scaloni, and E. Néron. Ordonnancement sur des machines identiques avec splitting et temps de préparation dépendant de la sequence. In *MOSIM '01 : conference francophone de modélisation et simulation*, 2001.

[Ser96] Paolo Serafini. Scheduling jobs on several machines with the job splitting property. *Operations Research*, 44(4) :617–628, 1996.

[TYCA06] Djamel Nait Tahar, Farouk Yalaoui, Chengbin Chu, and Lionel Amodeo. A linear programming approach for identical parallel machine scheduling with job splitting and sequence-dependent setup times. *International Journal of Production Economics*, 99(1) :63–73, 2006.

[Urr14] Javier Felipe Urrutia Jarpa. Modelo PAP que considera política de control de máxima demanda eléctrica en la producción de empresas manufactureras. Master's thesis, Universidad de Concepción, Facultad de Ingeniería, Departamento de Ingeniería Industrial, Chile, 2014.

[XZ00] Wenxun Xing and Jiawei Zhang. Parallel machine scheduling with splitting jobs. *Discrete Applied Mathematics*, 103(1) :259–269, 2000.

[XZZG13] Jing Xiao, Canrong Zhang, Li Zheng, and Jatinder ND Gupta. Mip-based fix-and-optimise algorithms for the parallel machine capacitated lot-sizing and scheduling problem. *International Journal of Production Research*, 51(16) :5011–5028, 2013.

[YC03] Farouk Yalaoui and Chegbin Chu. An efficient heuristic approach for parallel machine scheduling with job splitting and sequence-dependent setup times. *IIE Transactions*, 35(2) :183–190, 2003.

## Appendix

| | | MILP3 | | | | MILP2 | | | |
|---|---|---|---|---|---|---|---|---|---|
| instance | #batch | consump. | tard. | gap | time(s) | peak | tard. | consump. | gap |
| 1 | 2 | 15409 | 0 | 0 | 0,69 | 30 | 0 | 15341,2 | 49,44 |
| 1 | 3 | 15388,6 | 0 | 0 | 0,86 | 30 | 0 | 15388,6 | 49,6 |
| 1 | 4 | 15341,8 | 0 | 0 | 1,63 | 30 | 0 | 15341,8 | 100 |
| 2 | 2 | 17198 | 69,95 | 99,81 | 50 | 30 | 66,18 | 17207,4 | 82,69 |
| 2 | 3 | 17190 | 78,15 | 99,82 | 50 | 30 | 77,08 | 17192,7 | 91,38 |
| 2 | 4 | 17150,3 | 100,18 | 99,85 | 50 | 30 | 100,09 | 17150,6 | 96,26 |
| 3 | 2 | 16798,1 | 28,11 | 99,41 | 50 | 30 | 28,11 | 16798,1 | 71,95 |
| 3 | 3 | 16798,1 | 28,11 | 99,41 | 50 | 30 | 28,11 | 16798,1 | 82,73 |
| 3 | 4 | 16798,1 | 28,11 | 99,41 | 50 | 30 | 28,11 | 16798,1 | 99,35 |
| 4 | 2 | 17577,2 | 49,56 | 99,65 | 50 | 30 | 49,56 | 17465,9 | 78,76 |
| 4 | 3 | 17577,2 | 49,56 | 99,65 | 50 | 30 | 50,56 | 17522,2 | 88,13 |
| 4 | 4 | 17579,7 | 49,56 | 99,65 | 50 | 30 | 49,56 | 17557,2 | 100 |
| 5 | 2 | 16253,9 | 0 | 0 | 0,26 | 30 | 0 | 16253,9 | 48,33 |
| 5 | 3 | 16253,9 | 0 | 0 | 0,58 | 30 | 14,31 | 16254,2 | 64,97 |
| 5 | 4 | 16253,9 | 0 | 0 | 1,57 | X | X | X | X |
| 6 | 2 | 16624,3 | 0 | 0 | 0,28 | 30 | 0 | 16614,3 | 47,8 |
| 6 | 3 | 16624,3 | 0 | 0 | 1,37 | 30 | 0 | 16614,3 | 71,23 |
| 6 | 4 | 16624,3 | 0 | 0 | 1,35 | 30 | 0 | 16629,3 | 100 |
| 7 | 2 | 18604,5 | 7,45 | 97,58 | 50 | 30 | 3,42 | 18632,7 | 50,02 |
| 7 | 3 | 18576,2 | 15,62 | 98,83 | 50 | 30 | 9,44 | 18593,1 | 64,27 |
| 7 | 4 | 18576,2 | 13,99 | 98,69 | 50 | 30 | 10,16 | 18646,1 | 84,66 |
| 8 | 2 | 15227,3 | 0 | 0 | 0,36 | 30 | 0 | 15227,3 | 50,01 |
| 8 | 3 | 15222,3 | 0 | 0 | 1,04 | 30 | 0 | 15235,2 | 57,38 |
| 8 | 4 | 15131,9 | 0 | 0 | 1,57 | 30 | 0 | 15131,9 | 100 |

Table B.4 – Detailed results of experiments for (MILP2+MILP3).

**Auteur :** Margaux NATTAF

**Titre :** Ordonnancement sous contraintes d'énergie

**Directeur de thèse :** Christian ARTIGUES et Pierre LOPEZ

Soutenue le 18/10/2016 au LAAS-CNRS (Toulouse)

**Résumé :** Les problèmes d'ordonnancement à contraintes de ressource ont été largement étudiés dans la littérature. Cependant, dans la plupart des cas, il est supposé que les activités ont une durée fixe et nécessitent une quantité constante de la ressource durant toute leur exécution. Dans cette thèse, nous nous proposons de traiter un problème d'ordonnancement dans lequel les tâches ont une durée et un profil de consommation de ressource variables. Ce profil, qui peut varier en fonction du temps, est une variable de décision du problème dont dépend la durée de la tâche associée. Par ailleurs, la considération de fonctions de rendement linéaires et non linéaires pour la représentation de l'utilisation des ressources complexifie le problème et permet de modéliser de manière réaliste les transferts de ressources énergétiques. Pour ce problème NP-complet, nous présentons plusieurs propriétés permettant de dériver des modèles et méthodes de résolution. Ces méthodes de résolution sont divisées en deux parties. La première partie visualise ce problème du point de vue de la Programmation Par Contraintes et plusieurs méthodes dérivées de ce paradigme sont détaillées dont le développement du raisonnement énergétique sur le problème étudié. La seconde partie de la thèse est dédiée à des approches de Programmation Linéaire Mixte et plusieurs modèles, notamment un modèle à temps continu basé sur les événements, ainsi que des analyses théoriques et des techniques d'amélioration de ces modèles sont présentés. Enfin, des expérimentations viennent appuyer les résultats présentés dans ce manuscrit.

**Mots-clés :** ordonnancemment, énergie, recherche arborescente et locale, programmation mathématique, propagation de contraintes, complexité

**Discipline administrative :** Informatique

**Adresse du laboratoire :**

Laboratoire d'Analyse et d'Architecture des Systèmes

7, avenue du Colonel Roche

BP 54200

31031 Toulouse cedex 4, France