UNIVERSITÉ
FRANÇOIS RABELAIS
DE TOURS

École Doctorale M.I.P.T.I.S.

ÉQUIPE de RECHERCHE : Ordonnancement et Conduite

THÈSE en CO-TUTELLE avec l'Univertsité de Tunis

présentée par :

Anis KOOLI

soutenue le : 17 Juillet 2012

pour obtenir le grade de : Docteur de l'Université François Rabelais de Tours

Discipline/ Spécialité : Informatique

Exact and Heuristic Methods for Resource Constrained Project Scheduling Problem

Méthodes Exactes et Approchées pour le Problème de Gestion de Projet à Contraintes de Ressources

THÈSE DIRIGÉE PAR :

HAOUARI Mohamed	Professeur, Université de TUNIS
NÉRON Emmanuel	Professeur, Université François Rabelais de TOURS

JURY :

ARTIGUES Christian	Chargé de Recherche HDR, CNRS, TOULOUSE
HAOUARI Mohamed	Professeur, Université de TUNIS
LADHARI Talel	Professeur Associé, Université de TUNIS
LOPEZ Pierre	Directeur de Recherche CNRS, TOULOUSE
MOALLA Mohamed	Professeur, Université de TUNIS
NÉRON Emmanuel	Professeur, Université François Rabelais de TOURS

Acknowledgments

I would like to express my gratitude to my supervisors, Pr. Mohamed Haouari and Pr. Emmanuel Néron, for their encouragements, valuable guidance and their availability during my research and writing of my thesis. Also for their human qualities which helped me throughout the preparation of my research and make task easier and more enjoyable.

I would like to express my sincere thanks to Pr. Jacques Carlier for his advices and his scientific contribution.

I must also thank Dr. Lotfi Hidri for his support and great help during all the steps of preparation of my thesis.

Special Thanks are affectionaly aimed to my colleagues of the ROI and OC research groups for their friendship and their help.

Anis Kooli

Résumé

Le problème de gestion de projets à contraintes de ressources communément appelé Resource Constrained Project Scheduling Problem (RCPSP) est un problème central dans le domaine de la Recherche Opérationnelle et plus particulièrement dans la théorie de l'ordonnancement. Le problème consiste à ordonnancer plusieurs activités soumises entre elles à des contraintes de précédences et dont l'exécution requiert une ou plusieurs ressources. Chaque ressource à une capacité limitée et est disponible en permanence à partir du temps zéro. Le but est de trouver le temps de début de chaque activité qui permette de minimiser la date de fin du projet tout en respectant les contraintes de ressources et les contraintes de précédence.

Le RCPSP est un problème d'ordonnancement fondamental qui a fait l'objet d'études approfondies. De très nombreux travaux de la littérature ont proposé des méthodes de résolution pour ce problème : bornes inférieures, méthodes exactes, heuristiques et méta heuristiques.

Dans la première partie de la thèse, plusieurs types de bornes inférieures ont été proposés. Les premières bornes inférieures développées reposent sur la notion d'instance réduite. A partir d'une valeur triviale (calculée à partir de bornes usuelles comme la borne capacité, chemin critique, chemin critique amélioré ...), on calcule des temps de début au plus tôt et des temps de fin au plus tard pour les activités. De plus, l'horizon est subdivisé en plusieurs intervalles calculés à partir de ces instants. Une instance réduite est construite alors sur chaque intervalle en prenant comme temps d'exécution la partie obligatoire de chaque activité sur cet intervalle. Les relations de précédence et les consommations en ressources sont gardées telle quelles. A partir de cette instance réduite, les bornes usuelles peuvent être appliquées. Si la valeur de la borne calculée est supérieure à la largeur de l'intervalle alors une infaisabilité est détectée et la valeur de la borne peut être incrémentée (ou bien une recherche dichotomique peut être appliquée). Le second type de bornes inférieures est basé sur des améliorations du raisonnement énergétique classique. Ces améliorations sont réalisées par le biais d'extensions du raisonnement énergétique revisité initialement proposé pour le problème à machines parallèles. Ces améliorations essayent d'augmenter la valeur du travail sur chaque intervalle à travers des formulations mathématiques en nombres entiers. Une relaxation du raisonnement énergétique revisité a également été proposée donnant lieu à des modèles de problèmes de type sac-à-dos multidimensionnel.
Enfin, toutes ces bornes ont été améliorées en utilisant la notion de fonction duale réalisable qui permet de rajouter des ressources au problème initiale. Ceci a pour effet de détecter plus d'infaisabilités.

Ces différentes évaluations par défaut se sont avérées tout à fait compétitives avec les meilleures bornes inférieures de la littérature, et présentent un compromis temps de calcul/efficacité réellement intéressant. Des résultats expérimentaux ont été menés sur les instances du benchmark PSPLIB a permis d'identifier les bornes inférieures les plus pertinentes.
Ce travail a donné lieu à la publication d'un article dans la revue internationale *Computers and Operations Research*, une des revues les plus reconnues du domaine.

La deuxième partie du travail concernant l'évaluation par défaut de la durée optimale d'un projet, est basée sur une formulation linéaire déjà proposée dans la littérature. Cette formulation, peu efficace dans sa forme initiale, a été enrichie de nombreuses inégalités valides permettant par exemple de restreindre la préemption, de respecter autant que possible les contrainte de précédence entre les activités, d'introduire des incompatibilités dans l'exécution de plusieurs tâches simultanément par l'utilisation de cliques de disjonction etc. Là encore un travail expérimental minutieux a été entrepris en vue de d'identifier la configuration d'inégalités valides aboutissant au meilleur compromis efficacité/temps de calcul. Les bornes proposées permettent d'améliorer les meilleures bornes de la littérature dans un temps très compétitif : 48 bornes inférieures ont été améliorées dans les instances du Benchmark PSPLIB. De plus, ces bornes étant basées sur un découpage de l'horizon en intervalles successifs, elles peuvent être adaptées selon la taille du problème et permettent de calculer des bornes inférieures efficaces sur des grandes instances (jusqu'à 120 activités).

La dernière partie du travail concerne la proposition de nouvelles méthodes exactes de type Séparation/Evaluation pour le problème de gestion de projet à contraintes de ressources. Dans un premier temps, la meilleure borne inférieure développée dans la première partie de la thèse a été adaptée pour un schéma de branchement classique existant dans la littérature. Dans un second temps, un schéma de branchement original a été proposé basé sur le découpage de l'ordonnancement en plusieurs blocs. Les résultats expérimentaux montrent que seulement les projets à 30 activités peuvent être résolus. Malheureusement, comme pour les meilleures méthodes exactes de la littérature, la procédure ne permet pas de résoudre les problèmes à 60 activités. Si l'efficacité des méthodes proposées doit être améliorée, il est évident qu'il s'agit là d'un travail particulièrement difficile et constitue de ce fait un axe de recherche très intéressant.

Mots clés : Gestion de projets à contraintes de ressources, Raisonnement énergétique, Formulation mathématique, Borne inférieure, Procédure par séparation et évaluation, Règle de dominance.

Abstract

The Resource Constrained Project Scheduling Problem (RCPSP) is a central problem in the operations research field and particularly in the scheduling theory.

The problem is to schedule several activities subject to precedence constraints and whose execution requires one or more resources. Each resource is continuously available from time zero onwards and has a limited capacity. The objective is to find the start time of each activity that allows to minimize the end of the project (also called makespan) while respecting the resource and precedence constraints.

The RCPSP is a fundamental scheduling problem that has been widely studied in the literature. Numerous published works have proposed methods for solving this problem: lower bounds, exact methods, heuristics and meta heuristics.

In the first part of the thesis, several types of lower bounds have been proposed. The first lower bounds developed are based on the notion of reduced instance. From a trivial value (computed from simple lower bounds as the capacity bound, the critical path, the critical sequence ...), we calculate the release dates and the due dates of each activity. Moreover, the time horizon is divided into several intervals calculated from these latter instants. Then, a reduced instance is constructed on each interval by taking the mandatory part of each activity on this interval instead of its processing time. The precedence relations and resource demands are kept as is. From this reduced instance, simple lower bounds can be applied. If the calculated value of the lower bound is greater than the width of the time-interval then an infeasibility is detected and the value of the lower bound can be incremented (or a binary search can be applied).

The second type of lower bounds is based on improvements of the classical energetic reasoning. These improvements are realized through extensions of the revisited energetic reasoning initially proposed for the parallel machines problem with heads and tails. These improvements try to increase the value of the work on each interval through integer programming formulations. A relaxation of the revisited energetic reasoning has also been proposed resulting to multi-dimensional knapsack problems.

Finally, all these bounds were improved by using the concept of dual feasible functions that allows to add fictious resources to the initial problem. This results of detecting more infeasibilities.

The developed lower bounds proved to be quite competitive with the best lower bounds of the literature and present a compromise between efficiency and computation time which is really interesting. Experimental results were conducted on the PSPLIB benchmark instances in order to identify the most relevant lower bounds.

This work resulted in the publication of an article in the international journal *Computers and Operations Research*, one of the most recognized journals of the field.

The second part of the work concerning the evaluation of the optimal value of the makespan, is based on a linear formulation already proposed in the literature. This formulation, inefficient in its original form, was enriched with many valid inequalities in order to restrict preemption as much as possible, to respect the precedence constraints between activities, to introduce incompatibilities between activities by the use of cliques of disjunction etc. Again a careful experimental work was undertaken to identify the best configuration of valid inequalities leading to good trade off between efficiency and computational time. The proposed lower bounds improve the best lower bounds of the literature in a very competitive time: 48 lower bounds have been improved in the instances of the PSPLIB Benchmark. Moreover, these lower bounds are based on a subdivision of the horizon into successive intervals, can be adapted depending on the size of the problem and allow the calculation of effective lower bounds on large instances (up to 120 activities).

In the last part of the work, new Branch-and-Bounds procedures were proposed for the RCPSP. Initially, the best lower bound developed in the first part of the thesis has been adapted for an existing branching scheme. In a second step, an original branching scheme has been proposed based on the subdivision of the schedule into several blocs. The experimental results show that only 30-activity projects can be solved. Unfortunately, as the best exact methods proposed on the literature, the procedure can not solve the problems up to 60 activities. If the effectiveness of the proposed methods must be improved, it is clear that this is an extraordinarily difficult task and therefore constitutes a very interesting perspective of research.

Keywords : Resource Constrained Project Scheduling Problem, Energetic Reasoning, Mathematical formulation, Lower bound, Branch-and-Bound, Dominance rule.

Contents

Introduction **1**

1 The Project Scheduling Problem **3**

1.1 Introduction 3

1.2 Description of the RCPSP 3

1.2.1 Definition 3

1.2.2 RCPSP as a generalization of machine scheduling problems 5

1.3 Applications of the RCPSP 5

1.3.1 High-school timetabling 5

1.3.2 Cutting stock problem 6

1.4 Variants of the RCPSP 7

1.4.1 RCPSP with time lags 7

1.4.2 Multi-Mode project scheduling problem 8

1.4.3 Multi-Skill project scheduling problem 8

1.4.4 Alternative objective functions 9

1.5 Conclusion 10

2 Literature Review on the RCPSP **11**

2.1 Introduction 11

2.2 Mathematical formulation 11

2.3 Lower bounds 13

2.3.1 Combinatorial lower bounds 13

2.3.2 Linear programming based lower bounds 15

2.3.3 Destructive lower bounds 15

2.4 Exact procedures 16

2.4.1 Dynamic programming approach 16

2.4.2 Constraint programming 16

2.4.3 Branch-and-Bound procedures 17

2.5 Heuristics 18

2.5.1 Constructive heuristics 18
2.5.2 Metaheuristics 18
2.6 Benchmark instances 19
2.7 Conclusion 20

3 Enhanced Energetic Reasoning for the RCPSP 22
3.1 Introduction 22
3.2 Classical energetic reasoning 23
3.3 Improved energetic reasoning 24
3.4 Revisited energetic reasoning 27
3.4.1 Revisited energetic reasoning principle 28
3.4.2 Notation 29
3.4.3 Simple feasibility conditions 31
3.4.4 Exact revisited energetic reasoning 32
3.4.5 Relaxed revisited energetic reasoning 33
3.4.6 Global revisited energetic reasoning 35
3.5 Energetic reasoning and DFFs 37
3.6 Experimental results 38
3.6.1 First set of experiments: computational performance of the proposed lower bounds 39
3.6.2 Second set of experiments: assessment of the impact of DFFs . . . 41
3.6.3 Third set of experiments: performance of a shaving procedure . . 41
3.7 Conclusion 43

4 A Preemptive Bound for the RCPSP 45
4.1 Introduction 45
4.2 A basic preemptive formulation 46
4.3 A precedence-based cut 48
4.4 Incompatible activity sets-based cuts 50
4.4.1 Three cuts from the literature 50
4.4.2 Maximum number of activities in parallel 52
4.4.3 Clique cuts 53
4.4.4 Dual feasible functions-based cuts 54
4.5 Nonpreemptive cuts 55
4.5.1 First family 55
4.5.2 Second family 55
4.5.3 Third family 56
4.5.4 Fourth family 58

4.6 Energetic reasoning-based cuts 58
4.6.1 The concept of mandatory parts 58
4.6.2 A work-based cut 59
4.6.3 No idle-time 59
4.7 Experimental results 60
4.7.1 Performance of the proposed cuts 60
4.7.2 Comparison with lower bounds of Brucker and Knust [23] and Baptiste and Demassey [10] 62
4.7.3 Comparison with the approach of Schutt et al. [96] 63
4.8 Conclusion 64

5 Exact Methods for the RCPSP **65**
5.1 Introduction 65
5.2 The precedence tree branching scheme 65
5.2.1 Description of the scheme 65
5.2.2 Dominance rules 67
5.3 Forward-Backward branching schemes 69
5.3.1 Implicit enumerative search algorithm 70
5.3.2 Bloc-based branching scheme 71
5.3.3 Adaptation of the dominance rules 72
5.4 Experimental results 73
5.4.1 Impact of the priority rules 75
5.4.2 Impact of the energetic-based lower bound 75
5.4.3 Impact of the forward-backward schemes 76
5.5 Conclusion 76

Conclusion **78**

Appendices **81**

A New Lower Bounds **81**

List of Tables

3.1 Results of the bounds on KSD30 40
3.2 Results of the bounds on KSD60 40
3.3 Impact of DFFs on KSD30 41
3.4 Impact of DFFs on KSD60 41
3.5 Results of the shaving on KSD30 42
3.6 Results of the shaving on KSD60 42
3.7 Results of the shaving on KSD90 43
3.8 Results of the shaving on KSD120 43
3.9 Results of the bounds on BL instances 43
3.10 Results of the bounds on PACK instances 44

4.1 Comparison between LB_{CN} and LB_{BK} on nontrivial instances 48
4.2 Results of the bounds on 114 non trivial KSD30 instances 61
4.3 Results of the bounds on 118 non trivial KSD60 instances 61
4.4 Comparison of the bounds on KSD instances 62
4.5 Comparison between LB_{Pmtn}, LB_{BK} and LB_{BD} on KSD instances 63
4.6 Comparison between LB_{Pmtn} and LB_{CP} on KSD instances 63

5.1 Impact of the priority rules 75
5.2 Impact of the energetic-based lower bound 76
5.3 Impact of the FBBS 76

A.1 New lower bounds for KSD90 instances 81
A.2 New lower bounds for KSD120 instances 82

List of Figures

1.1 The activity-on-node network for Example 1 5
1.2 Two feasible schedules for Example 1 6
1.3 Solution of a cutting-stock problem 7

3.1 Improved energetic reasoning project example 26
3.2 Reduced instance on time-interval [7,14] 27
3.3 Classical energetic reasoning principle 28
3.4 Revisited energetic reasoning principle 29
3.5 Global revisited energetic reasoning project example 36

4.1 Illustration of nonpreemptive cuts (second family) 56
4.2 Example of nonpreemptive cuts (third family) 57

5.1 Exact methods instance example 66
5.2 Search tree produced by the PTBS 67
5.3 A schedule obtained by the forward-backward scheme 69
5.4 Decomposition of a schedule into blocs 71
5.5 Optimal schedule obtained by the bloc-based scheme 72

Introduction

In real life situations, decisions to be made are often constrained by specific requirements which are typically conflicting in nature. The decision making process gets increasingly more complicated with increment in the number of constraints. Modeling and development of solution methodologies for these scenarios have been the challenge for operations researchers from the outset.

In such a rough economic environment, the key functions of the success for the industrial companies lie in their abilities to produce the required items at the right time and with the lowest possible cost. One of the main key functions to satisfy this harsh constraints are the production planning and scheduling.

The problem of project scheduling subject to resource constraints is one of the problems particularly studied in recent years by many researchers dealing with problems of Operations Research and specifically, scheduling [5]. Because of its theoretical and practical interest, project scheduling problems received a great attention since the early 80s.

A project consists of a number of tasks subject to precedence constraints and whose execution requires one or more resources. These resources can be of different types (Human, physical, monetary, etc.) and are available in limited quantities. The goal is to find for each task a start date that minimizes the test considered, e.g., the date of completion of the project, the largest delay, etc. while respecting the resources and precedence constraints. Besides some theoretical interest, industrial applications based on project scheduling models are numerous. We refer to surveys, e.g., [5, 42, 60, 56] for a presentation of several of these applications, for which resolution methods are based largely on traditional methods of literature.

In this thesis, we focus on the classical version of the Resource Constrained Project Scheduling Problem (RCPSP). In the standard RCPSP, the problem consists in scheduling activities submitted to both precedence relationship and resource constraints, in order to minimize the project duration. Optimizing the project duration becomes a crucial point in most of the organizations, such as engineering, manufacturing systems and Research and Development projects.

Lot of lower bounds have been proposed for the RCPSP. Lower bounds are extremely useful both from practical and theoretical point of view, especially for evaluating the quality of feasible solutions. In addition the most efficient exact procedures [41, 72] proposed in the literature can only handle small sized instances of 30 activities. This is why researchers turned to heuristic methods to tackle the problem. In [65], the authors compare more than 250 heuristic procedures.

This thesis work consists in developing lower bounds and exact methods for the RCPSP. The thesis is composed of five chapters. In the following, we give a brief overview of the content of each chapter.

- **Chapter 1**, presents the RCPSP. We give a description of the problem and some of its properties. Possible variants of the standard problem are presented.
- **Chapter 2**, presents a survey of the RCPSP. First, we present two mathematical formulations of the problem. Then, we describe, the different approaches to solve the problem. We begin by presenting some lower bounds. Then, we describe exact procedures. The chapter is concluded by presenting heuristic methods used to solve approximately the problem.
- **Chapter 3**, provides new lower bounds for the RCPSP based on the concept of *energetic reasoning*. We propose new efficient approaches that improves the classical energetic reasoning procedure. We also use the notion of *Dual Feasible Functions* in order to tighten the feasibility conditions and adjustments. Further numerical experiments analysis are proposed.
- **Chapter 4**, is dedicated to a preemptive relaxation based LP formulation. In the first part of the chapter, we present a basic preemptive formulation. Then we present all the improvements that we propose to strengthen this relaxation. Finally, computational experiments on classical benchmark are presented.
- **Chapter 5**, presents exact procedures for the RCPSP. We begin by adapting the developed lower bounds to a classical branching scheme. Then, we propose a new branching scheme which consists in placing, alternatively, the activities at the beginning and the end of the schedule. Numerical experiments are conducted to test the efficiency of the procedures.

Chapter 1

The Resource Constrained Project Scheduling Problem

1.1 Introduction

The resource constrained project scheduling problem (**RCPSP**) is a very general scheduling problem which may be used to model many applications in practice (e.g. a production process, a software project, a school timetable, the construction of a house or the renovation of an airport). The objective is to schedule some activities over time such that scarce resource capacities are respected and a certain objective function is optimized. Examples for resources may be machines, people, rooms, money or energy, which are only available with limited capacities. As objective functions e.g. the project duration, the deviation from deadlines or costs concerning resources may be minimized. In this chapter, we introduce the basic version of the problem and some possible extensions. In Section 1.2, we describe the RCPSP and introduce essential notation. Then, we present in Section 1.3 two applications of the RCPSP. Finally, we dedicate Section 1.4 to present various extensions and generalizations of the classical problem.

1.2 Description of the RCPSP

1.2.1 Definition

The RCPSP is a central problem in scheduling theory that has great relevance in project management and more specifically to the crucial issue of allocating scarce resources to activities. Formally, the RCPSP is defined as follows: We consider a project that consists of a set $\mathcal{A}$ of n activities to be scheduled nonpreemptively. A set $\mathcal{R}$ of K renewable resources are required for processing these activities. Each resource k ($k \in \mathcal{R}$) is continuously available from time zero onwards with resource capacity B_k. The processing of an activity j ($j \in \mathcal{A}$) lasts p_j units of time, and requires b_{jk} units of resource k ($k \in \mathcal{R}$). Moreover, the activities are interrelated through precedence constraints. These time restrictions are often modeled using an appropriate activity-on-node graph

$G = (V, A)$. In this graph, $V = \{0, 1, \ldots, n+1\}$ is the set of nodes corresponding to the n activities, in addition to two dummy zero-duration activities 0 and $n+1$ that represent the start and the end of the project, respectively. Each vertex $j \in V$ is weighted with the corresponding processing time p_j. The dummy activities need no resources and have processing time zero. For all activities j ($j \in \mathcal{A}$), we set 0 precedes j for all activities j without any predecessor and j precedes $n+1$ for all activities j without any successor. The arcset A represents the precedence restrictions (that is, $(i, j) \in A \Leftrightarrow i$ is an immediate predecessor of j).

The objective is to determine starting times s_j ($j \in \mathcal{A}$) for the activities in such a way that:

- at each time t the total resource demand is less than or equal to the resource availability of each resource k ($k \in \mathcal{R}$),
- the given precedence constraints are fulfilled, i.e., $s_i + p_i \leq s_j$ if i precedes j, and
- the makespan $C_{max} = \max_{j=1}^{n} C_j$ is minimized, where $C_j = s_j + p_j$ is assumed to be the completion time of activity j.

It is well-known that the RCPSP is strongly $\mathcal{NP}$-hard (Blazewicz et al. [18]). Added to that and from a computational perspective, the RCPSP turns out to be an extremely hard nut to crack. Indeed, to give an insight of the notorious intractability of the RCPSP, we mention that state-of-the-art exact algorithms fail to solve some well-known 60-activity benchmark instances that are still open about fifteen years after their publication (see Artigues et al. [5]).

The vector $S = (s_j)$ defines a schedule of the project. Vector S is called feasible if all resource and precedence constraints are fulfilled.

Moreover, for each activity j, we define:

$$Pred(j) = \{i \in \mathcal{A} : (i, j) \in A\} \text{ and } Succ(j) = \{i \in \mathcal{A} : (j, i) \in A\}$$

as the sets of predecessors and successors of activity j ($j \in \mathcal{A}$), respectively.

Example 1. *Consider a project with $n = 4$ activities, $K = 2$ resources with capacities $B_1 = 5$, $B_2 = 7$. In addition, we suppose that activity 2 precedes activity 3 and the following data:*

j	1	2	3	4
p_j	4	3	5	8
b_{j1}	2	1	2	2
b_{j2}	3	5	2	4

*Figure 1.1 illustrates the corresponding activity-on-node network, where the dummy activities 0 and 5 have been added and the vertices are weighted with the processing times. In Figure 1.2(a) a so-called **Gantt chart** of a feasible schedule with $C_{max} = 15$ is drawn. This schedule does not minimize the makespan, since by moving activity 1 to the right, a shorter schedule is obtained. An optimal schedule with makespan $C_{max} = 12$ is shown in Figure 1.2(b).*

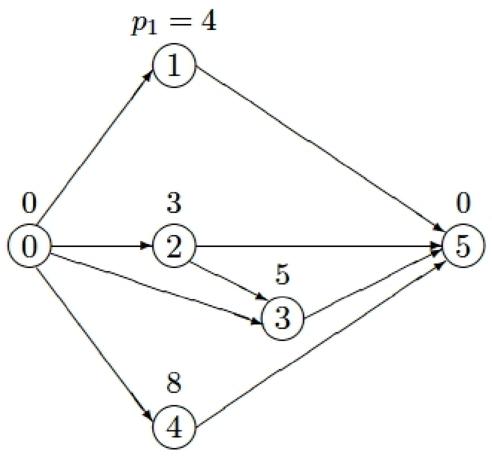


Figure 1.1: The activity-on-node network for Example 1

1.2.2 RCPSP as a generalization of machine scheduling problems

The RCPSP is a generalization of machine scheduling problems such as single machine, parallel machine and shop problems. For the single machine scheduling problem, we are given n jobs $j = 1, \ldots, n$ with processing times p_j which have to be processed on a single machine. Such a problem can be modeled by an RCPSP with one renewable resource with capacity $B_1 = 1$ and resource requirements $b_{j1} = 1$, $\forall j = 1, \ldots, n$.
For the parallel machine problem, we have m identical machines (the processing time p_j of a job does not depend on the machine on which the job is processed) which can work simultaneously. We can model this problem with a special RCPSP with one resource with capacity $B_1 = m$. The resource demands are the same as for the single machine case.

1.3 Applications of the RCPSP

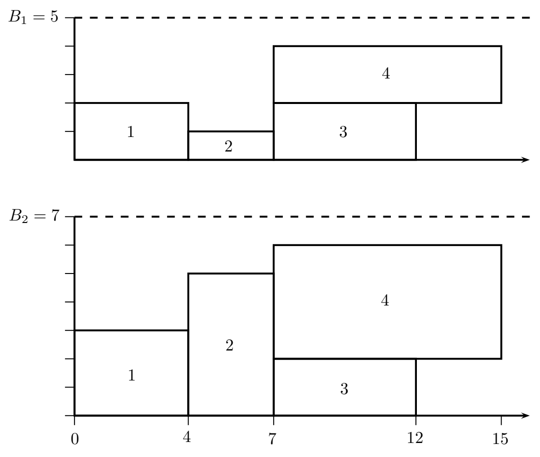
In this paragraph, we describe two simple applications of the RCPSP. We refer to the book of Brucker and Knust [24] for further details on these applications and other ones.

1.3.1 High-school timetabling

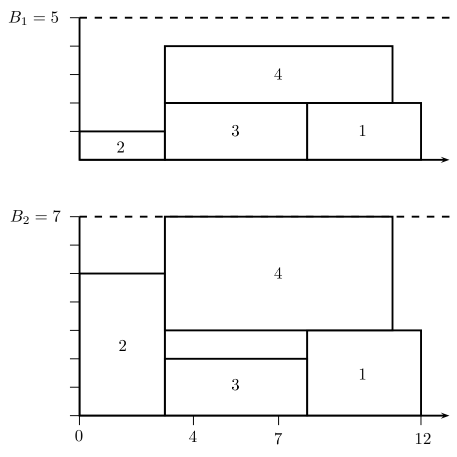
In a basic high-school timetabling problem we are given M classes $C_1, \ldots, C_M$, L teachers $H_1, \ldots, H_L$ and P teaching periods $T_1, \ldots, T_P$. Furthermore, we have a set of n lectures $A_1, \ldots, A_n$. Associated with each lecture is a unique teacher and a unique class. In order to simplify the presentation, we suppose that all teachers are available in all teaching periods.

The corresponding timetabling problem is to assign the lectures to the teaching periods such that:

- each class has at most one lecture in any time period,



(a) A feasible schedule



(b) An optimal schedule

Figure 1.2: Two feasible schedules for Example 1

- each teacher has at most one lecture in any time period.

This problem may be formulated as an RCPSP with n activities, where each activity corresponds to a lecture given by some teacher for some class. Furthermore, we have $K = M + L$ resources. The first M resources correspond to the classes $C_1, \dots, C_M$ and the last L resources correspond to the teachers $H_1, \dots, H_L$. We have $B_k = 1, \forall k = 1, \dots, M + L$.
If activity A_j is a lecture for class C_m given by teacher H_l , then its resource requirement for resource k is given by:

$$b_{jk} = \begin{cases} 1 & \text{if } k = m \text{ or } k = M + l \\ 0 & \text{otherwise} \end{cases}$$

In a basic version of the problem one has to find a feasible schedule with $C_{max} \leq T$. In practice, many additional constraints may have to be satisfied, e.g.

- for each class or teacher the number of teaching periods per day is bounded,
- certain lectures must be taught in special rooms,
- some pairs of lectures have to be scheduled simultaneously, etc.

1.3.2 Cutting stock problem

Materials such as paper, textiles, cellophane, etc. are manufactured in standard rolls of a large width W which is the same for all rolls. These rolls have to be cut into smaller rolls j, $j = 1, \dots, n$ with widths w_j such that the number of sliced rolls is minimized. In Figure 1.3 a solution of a cutting-stock problem with $n = 15$ smaller rolls is illustrated. In this solution 7 rolls have to be cut.

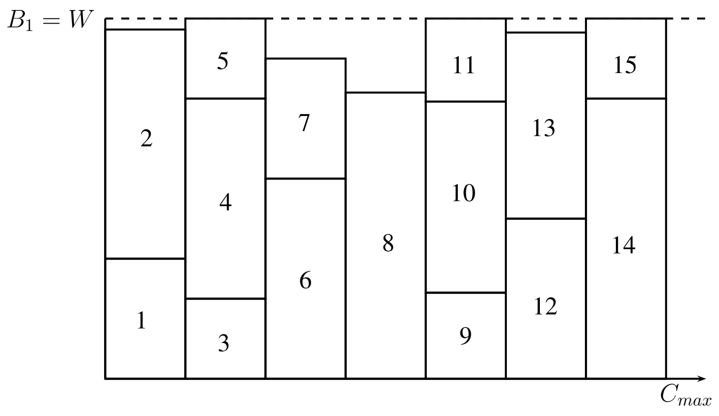


Figure 1.3: Solution of a cutting-stock problem

This problem can be formulated as an RCPSP with only one renewable resource with capacity $B_1 = W$ units. The activities j correspond to the rolls to be cut. Activity j has processing time $p_j = 1$ and uses $b_{j1} = w_j$ units of this resource. The makespan corresponds to the number of standard rolls to be cut, i.e., a schedule with a minimal makespan corresponds to a solution with a minimal number of sliced standard rolls.

1.4 Variants of the RCPSP

In the following, we will discuss different variants of the standard RCPSP. We refer to the paper of Hartmann and Briskorn [56] for a survey of variants and extensions of project scheduling problems.

1.4.1 RCPSP with time lags

For this problem (usually denoted by **RCPSP/max**), in addition to the precedence relations, some minimal and maximal start-start time-lags (generalized precedence relations) between the activities are given. If we consider activity i and activity j, a distance d_{ij} has to be satisfied between i and j, i.e., the starting times s_i and s_j have to fulfill $s_j - s_i \geq d_{ij}$. If $d_{ij} \geq 0$, activity j cannot start earlier than d_{ij} time units after the start of activity i (minimal time-lag). On the other hand, if $d_{ij} < 0$, activity j cannot start earlier than $|d_{ij}|$ time units before the starting time of activity i, or, equivalently, activity i cannot start later than $|d_{ij}|$ time units after the start of activity j (maximal time-lag).
Bartusch et al. [13], De Reyck et al. [93], Dorndorf et al. [43] and Fest et al. [48] presented branch-and-bound algorithms, while Heilmann and Schwindt [57], Möhring et al. [79], Brucker and Knust [22] and Bianco and Caramia [17] calculated lower bounds. Heuristic solutions have been presented in Cesta et al. [34] and Neumann and Zimmermann [85].

1.4.2 Multi-Mode project scheduling problem

In the **multi-mode** case a set $\mathcal{M} = \{M_1, \ldots, M_V\}$ of so-called modes (processing alternatives) is associated with each activity j ($j \in \mathcal{A}$). The processing time of activity j in mode M_v is given by p_{jv} and the per period usage of a renewable resource k is given by b_{jkv}. One has to assign a mode to each activity and to schedule the activities in the assigned modes.
Multiple modes may for example be used in order to model a situation in which an activity can be rapidly processed by many workers or more slowly with less people.
This problem have been widely studied in the literature. Sprecher and Drexl [99] presented a branch-and bound algorithm to solve exactly the problem, while Boctor [19], Hartmann [53, 54], Bouleimen and Lecocq [20], Nonobe and Ibaraki [86], Maniezzo and Mingozzi [76], Alcaraz et al. [2] and Jarboui et al. [59] calculated heuristic solutions. Recently, Coelho and Vanhoucke [36] proposed a new algorithm which splits the problem type into a mode assignment and a single mode project scheduling step. The mode assignment step is solved by a satisfiability (SAT) problem solver and returns a feasible mode selection to the project scheduling step. The project scheduling step is then solved using an efficient metaheuristic procedure. Additionally, in [76, 22] lower bounds are calculated based on a linear programming formulation.

For more informations, we refer to the paper of Weglarz et al. [110] for a survey of the Multi-Mode problem and its extensions.

1.4.3 Multi-Skill project scheduling problem

In Multi-Skill project scheduling problem, the resources are staff members. Each member H_m, $m \in \{1, \ldots, M\}$, masters one or more specific skill(s) among all the skills K_l, $l \in \{1, \ldots, L\}$ existing in the project. Thus, each unit of skill required by an activity corresponds to a person that has to be assigned to do the required skill for this activity. For each activity $j \in \mathcal{A}$ and each skill K_l, a_{jl} is the number of persons that we have to assign to j to do K_l during the whole processing time of j. A person can be assigned to a need only if he/she masters the required skill. The objectif is to find a solution that minimize the makespan of the project.
To the best of our knowledge, contributions proposed for multi-skill project scheduling problem are very few: there are two different ways to take into account skills of employees, either the problem is to find a solution where the assignments match the skills of employees [15, 16] and eventually their level of abilities [14], or the problem is to compute a solution at a minimum cost under different constraints [94]. In the latter case, every assignment of an employee has a cost that grows up if the employee is not well skilled for the activity to do, moreover the global project has a due date, and there is a penalty if the project is delayed after this due date.

There is a link between the Multi-Mode Project Scheduling Problem (**MM-RCPSP**) and Multi-Skill Project Scheduling Problem (**MS-RCPSP**) models because MM-RCPSP formulation can be used to describe a MS-RCPSP instance. In fact, a mode corresponds to a given subset of staff members that matches the requirements of the activity. Every mode has the same processing time, and there exist as many different modes as feasible subsets of staff members satisfying needs of the activity. The main difference between MM-RCPSP and MS-RCPSP lies in the number of modes usually proposed for each activity: in classical instances of Multi-Mode RCPSP [66], there are at most 10 modes per activity, but if we want to enumerate the number of possible subsets of staff members in an instance of MS-RCPSP, it will be very much larger. In some small instances with 3 skills and 10 persons, the number of modes per activity can exceed 1000. Moreover, most of the exact methods proposed for solving exactly MM-RCPSP have a branching scheme based on an explicit enumeration of the modes for each activity. Thus, these methods cannot be used for solving exactly the MS-RCPSP.

1.4.4 Alternative objective functions

1.4.4.1 Time-oriented objective functions

Besides the objective of minimizing the makespan C_{max} one may consider other objective functions $f(C_1, \dots, C_n)$ depending on the completion times of the activities. Examples are the total flow time $\sum_{j=1}^{n} C_j$ or more generally the weighted (total) flow time $\sum_{j=1}^{n} w_j C_j$ with nonnegative weights w_j. Other objective functions depend on due dates d_j which are associated with the activities. With the lateness $L_j = C_j - d_j$ and the tardiness $Tj = \max\{0, C_j - d_j\}$, the following objective functions are common:

the maximum lateness	$L_{max} = \max_{j=1}^{n} L_j$
the total tardiness	$\sum_{j=1}^{n} T_j$
the total weighted tardiness	$\sum_{j=1}^{n} w_j T_j$

Nazareth et al. [82] and Nudtasomboon and Randhawa [88] propose to minimize the sum of all activity completion times, while Rom et al. [95] minimize the weighted sum of the completion times. Ballestín et al. [8], Kolisch [64] and Viana and de Sousa [108] consider the minimization of the weighted tardiness. Neumann et al. [83] describe the minimization of the maximum lateness.

1.4.4.2 Resource-based objective functions

Besides the time-oriented objective functions, resource-based ones may be considered. They occur for example in the area of resource investment and resource leveling problems.

In the *resource investment problem (RIP)* the resource capacities B_k are not given, but have to be determined as additional decision variables. Providing one unit of resource k costs $c_k \geq 0$. The objective is to find a schedule with $C_{max} \leq T$ for a given deadline

T where the resource investment costs $\sum_{k=1}^{K} c_k B_k$ are minimized. The resource investment problem has recently been tackled by Drexl and Kimms [44], Neumann and Zimmermann [84], Neumann et al. [83], Ranjbar et al. [92] and Yamashita et al. [112].

In *resource leveling problems (RLP)* the variation of the resource usage over time is measured. Let $c_k \geq 0$ be a cost for resource k and denote by $b_k^S(t)$ the resource usage of resource k in period $t \in \{1, \ldots, T\}$ for a given schedule S, where $b_k^S(0) = 0$ is assumed. Besides the resource capacity k a target value $Y_k \geq 0$ for resource k is given. In so-called *deviation* problems the deviations (overloads) of the resource usages from a given resource profile are minimized. This can be done by minimizing:

$$\begin{array}{ll} \text{the deviation} & \sum_{k=1}^{K} c_k \sum_{t=1}^{T} |b_k^S(t) - Y_k|, \\ \text{the overload} & \sum_{k=1}^{K} c_k \sum_{t=1}^{T} \max\{0, b_k^S(t) - Y_k\}, \\ \text{the squared deviation} & \sum_{k=1}^{K} c_k \sum_{t=1}^{T} (b_k^S(t) - Y_k)^2. \end{array}$$

Recently, Neumann and Zimmermann [84], Neumann et al. [83] and Nudtasomboon and Randhawa [88] have tackled resource leveling problems.

On the other hand, in so-called *variation* problems, the resource usages should not vary much over time. This can be achieved by minimizing one of the objective functions:

$$\sum_{k=1}^{K} c_k \sum_{t=1}^{T} |b_k^S(t) - b_k^S(t-1)|,$$

$$\sum_{k=1}^{K} c_k \sum_{t=1}^{T} \max\{0, b_k^S(t) - b_k^S(t-1)\},$$

$$\sum_{k=1}^{K} c_k \sum_{t=1}^{T} (b_k^S(t) - b_k^S(t-1))^2.$$

1.5 Conclusion

In this chapter, we presented the resource constrained project scheduling problem. We showed the importance of this problem from theoretical and practical points of view. We also presented some variants and extensions of the standard problem. These extensions are motivated by real life applications. In the next chapter, we review exact and heuristic methods proposed in the literature to solve the resource constrained project scheduling problem.

Chapter 2

Literature Review on the Resource Constrained Project Scheduling Problem

2.1 Introduction

The RCPSP is a fundamental scheduling problem that has been extensively investigated in the operations research literature. We refer to the excellent book by Demeulemeester and Herroelen [42] for a comprehensive review of the impressive research effort that has been devoted to the study of the RCPSP. In this chapter, we present a brief state of the art on resolution methods to solve this problem. In Section 2.2, we present a mathematical formulation modeling the RCPSP, followed by the lower bounding approaches in Section 2.3. After that, in Sections 2.4 and 2.5, we present the heuristic and exact methods for solving the problem, respectively, to finish, in Section 2.6, with an introduction of the benchmark instances that will be used for numerical tests in the next chapters.

2.2 Mathematical formulation

Lot of mathematical formulations were proposed to model the RCPSP. Roughly, these formulations can be classified into two families. The first family of formulations looks to the position of activities compared to other ones (see Alvarez-Valdés and Tamarit [3], Artigues [6], etc.). In the second family, the time horizon is subdivided into unitary time points and the starting times of the activities are determined. One of the first mathematical models falling into this category was proposed by Pritsker et al. [91]. The formulation uses time indexed variables x_{jt} where x_{jt} is equal to 1 if activity j starts at time t and 0 otherwise. The model can be presented as the following:

$$\min \sum_{t=0}^{T} t\, x_{n+1,t} \tag{2.1}$$

s.t.

$$\sum_{t\in[0,T]} x_{jt} = 1, \quad \forall j \in \mathcal{A} \tag{2.2}$$

$$\sum_{t=0}^{T} t(x_{jt} - x_{it}) \geq p_i, \quad \forall (i,j) \in A \tag{2.3}$$

$$\sum_{j\in\mathcal{A}} b_{jk} \sum_{s=t-p_j+1}^{T} x_{js} \leq B_k, \quad \forall k \in \mathcal{R},\, t \in [0,T] \tag{2.4}$$

$$x_{jt} \in \{0,1\}, \quad \forall j \in \mathcal{A},\, t \in [0,T] \tag{2.5}$$

Constraints (2.2) state that each activity is started exactly one time over the planning horizon T. Inequalities (2.3) impose that the precedence constraints must be satisfied. Inequalities (2.4) state that there is no resource conflicts at each time instant. Constraints (2.5) are the integrality constraints.

Later, Christofides et al [35] proposed several cuts in order to strengthen the linear relaxation of the mathematical model of Pritsker et al. [91]. The first cut is a modified version of the precedence constraints by presenting them in a *disaggregated* manner. This constraint is:

$$\sum_{s=t}^{T} x_{is} + \sum_{s=0}^{t+p_i-1} x_{js} \leq 1, \quad \forall (i,j) \in A, t \in [0,T] \tag{2.6}$$

To check that the previous constraint is valid, assume that i precedes j. Then, if i starts at time t then the earliest start time of j is $t+p_i$. Therefore, j cannot start at time $t+p_i-1$ or earlier. Hence, we have $y_{it} = 1 \Rightarrow \sum_{s=0}^{t+p_i-1} x_{js} = 0$. It follows that:

$$x_{it} + \sum_{s=0}^{t+p_i-1} x_{js} \leq 1$$

is a valid inequality. This inequality can be improved by observing that if the starting time of i is greater or equal to t, then $\sum_{s=0}^{t+p_i-1} x_{js} = 0$ as well. Hence, we have the disaggregated precedence constraint.

Another cut is the *clique* cut which is written as the following:

$$\sum_{j\in Cl} x_{jt} \leq 1, \quad \forall t \in [0,T] \tag{2.7}$$

where Cl is a set containing activities that cannot be in execution simultaneously at a time t.

2.3 Lower bounds

A lower bound is a value known to be less than or equal to the optimal value. A bound is *tight* if it is known to be close to the optimal value. The procedure followed in order to generate a lower bound can be termed a *bounding procedure*, though is often simply referred to as a *lower bound*. There are two primary uses of bounds in scheduling research. Lower bounds can be used to gauge the performance of a non-optimal algorithm by establishing the approximate 'position' of the optimal value, in the absence of a true optimal value. Lower bounds can also be used to guide a solution process, for example in *branch-and-bound algorithms*.

Lot of lower bounds have been proposed in the literature for the RCPSP. Roughly, these bounds can be classified into three families: combinatorial lower bounds, linear programming based lower bounds and destructive ones. In the following, we describe each approach.

2.3.1 Combinatorial lower bounds

Constructive lower bounds are mainly based on construction of optimal solution for relaxed versions of the initial problem. Relaxation can be on the resource constraints and/or precedence constraints. Generally, these kind of bounds need only few computational time, but the gap to optimal solution may be large.

In the following and for the sake of brevity, we restrict our attention to the lower bounds that we shall use in the subsequent chapters. For a comprehensive and thorough survey of lower bounds for the RCPSP, the reader is referred to [5].

2.3.1.1 The capacity bound

To begin with, we introduce a simple $O(Kn)$ bound that is often referred to as the *capacity bound*, and that is based on a relaxation of the precedence constraints while considering each resource separately. A bound value is computed as the total requirement of this resource divided by its per period availability and rounded up to the next larger integer. Formally, this lower bound is computed as follows:

$$LB_C = \max_{k \in \mathcal{R}} \left\lceil \sum_{j \in \mathcal{A}} \frac{b_{jk} p_j}{B_k} \right\rceil$$

2.3.1.2 The critical path bound

The *critical path bound* is a second simple lower bound that is based on a relaxation of the capacity constraints. The critical path corresponds to the longest path from 0 to $n+1$ in the precedence graph G. We define $l^*(j,s)$ as the longest path in the precedence graph between activity j and activity s. The release date r_j of activity j ($j = 0, \ldots, n+1$) is defined as $l^*(0,j)$. According to these definitions, the critical path value LB_{CP} is equal to r_{n+1}.

Symmetrically, let C be a trial value, then the deadline d_j of activity j ($j = 0, \ldots, n+1$) is defined as $d_j = C - l^*(j, n+1) + p_j$. Hence, a time-window $[r_j, d_j]$ is associated with each activity j.

2.3.1.3 The critical sequence bound

Stinson et al. [102] has proposed an improvement of the critical path lower bound. Activities that cannot be processed simultaneously, due to at least one resource constraint, are identified. First, the critical path CP is computed. Next, if an activity $j \notin CP$ cannot be processed simultaneously with $i \in CP$, then e^i_j the amount of time during which the two activities cannot be processed simultaneously is computed (according to both critical path and time-windows of activity). Then, $e^{min}_j = \min\limits_{i \in CP} e^i_j$ is computed which is the minimal increasing of the LB_{CP} value in order not to violate resource constraints. A valid lower bound, hereafter referred to as the *critical sequence bound*, is:

$$LB_{CS} = LB_{CP} + \max_{j \in \mathcal{A} \setminus CP} e^{min}_j$$

The complexity of LB_{CS} is $O(n^2)$.

2.3.1.4 The *m*-machine bound

The m-machine lower bound (Carlier and Latapie [27]) is based on a relaxation of the RCPSP as a parallel machine problem with heads and tails (that is, $P|r_j, q_j|C_{max}$). A relaxed instance is derived as follows:

- Let m be the number of machines ($m \in [1, \bar{m}]$ is a parameter and $\bar{m}$ represents the maximal number of machines. In our experiments, we empirically set $\bar{m} = 5$). For each resource k ($k \in \mathcal{R}$), we set $u^m_k = \left\lfloor \dfrac{B_k}{m+1} \right\rfloor + 1$,
- Let $\mathcal{E}^m_k = \{j \in \mathcal{A} : b_{jk} \geq u^m_k\}$ be the set of jobs to be scheduled,
- For each job $j \in \mathcal{E}^m_k$, $a_{jk} = \left\lfloor \dfrac{b_{jk}}{u^m_k} \right\rfloor$ is the number of jobs associated with j. All these a_{jk} jobs have a processing time p_j, a head r_j and a tail $q_j = d_{n+1} - d_j$. Let $\mathcal{S}^m_k$ be the set containing all the jobs associated to jobs $j \in \mathcal{E}^m_k$.

Thus, a $P|r_j, q_j|C_{max}$ instance is obtained. A valid $O(n \ln n)$ bound for this problem is:

$$LB_P(k, m) = \left\lceil \frac{1}{m} \Big(\bar{r}_1 + \ldots + \bar{r}_m + \sum_{j \in \mathcal{S}^m_k} p_j + \bar{q}_1 + \ldots + \bar{q}_m \Big) \right\rceil$$

where $\bar{r}_i$ and $\bar{q}_i$ represent the i^{th} smallest heads and tails of the activities of the set $\mathcal{S}^m_k$, respectively.

Hence, a RCPSP valid bound is:

$$LB_P = \max_{k \in \mathcal{R}} \max_{m \in [1, \bar{m}]} \{LB_P(k, m)\}\}.$$

In the sequel, we refer to LB_P as *the m-machine bound*.

2.3.2 Linear programming based lower bounds

The second family of lower bounds includes bounds that use linear relaxations or Lagrangean ones.

LP relaxation based approaches have been presented by Christofides et al. [35] , Mingozzi et al. [78] as well as Brucker et al. [25]. The latter two consider a mathematical model based on feasible subsets. Its LP relaxation corresponds to allowing preemption of jobs. Since the number of feasible subsets grows exponentially with the number of jobs, the computational effort is very high, though the quality of bounds obtained is good. Determining all feasible subsets in advance can be avoided by using the technique of column generation for solving the RCPSP with preemption as proposed in Weglarz et al. [109]. Carlier and Néron [28] also proposed lower bounds based on LP formulations which take into account how resource requirements can be satisfied simultaneously for a given resource capacity. Recently, Koné et al. [67] proposed two event based formulations for the RCPSP. They compared the linear relaxations of these new mathematical models with other formulations already proposed in the literature.

Lagrangean relaxation based approaches can be found in [49, 35, 38]. Christofides et al. [35] used the formulation of Pritsker et al. [91] with the new proposed precedence constraint (see paragraph 2.2) to derive their lower bound. This method was improved by Möhring et al. [80] by solving a minimum cut problem.

2.3.3 Destructive lower bounds

Symmetrically to constructive lower bounds that are generally computed at low computational effort, but may remains far from the optimal solution, destructive lower bounds based on ILP formulation have been proposed in the last years. Destructive lower bounds try to detect contradiction for a decision variant of the initial problem. The feasibility (decision) problem may be formulated as follows: Given a threshold value C, does a feasible schedule exists with an objective value smaller than or equal to C? If a contradiction is proved, then $C + 1$ is a valid lower bound for the optimization problem. To contradict (destruct) a threshold value C, again relaxations may be used. If we can state infeasibility for a relaxed problem, obviously the original problem is also infeasible. To find the best lower bound we search for the largest C, where infeasibility can be proved. To achieve this goal, a binary search on C can be performed to compute the lower bound.

This approach have been introduced by Klein and Scholl [63]. The authors propose several feasibility tests to compute lower bounds. Brucker and Knust [23] have proposed a LP based lower bound derived from the ILP formulation proposed by Mingozzi et al. [78]. The authors proposed an improvement of a lower bound by subdividing the time horizon into time intervals and using the technique of column generation to deal with the great number of variables. This formulation has been improved by adding a preprocessing phase (time-windows reduction) and valid inequalities by Baptiste and Demassey [10]. Furthermore, an additional approach using column generation was proposed by van den Akker et al. [107]. This approach requires reformulating the RCPSP into an equivalent problem of minimizing the maximum lateness on a set of identical parallel machines. Demassey et al. [39] also proposed several lower bounds based on linear relaxations of the ILP formulations of Pritsker et al.[91], Christofides et al. [35] in addition to a formulation proposed by Alvarez-Valdés et al. [3]. Finally, let us cite the work of Carlier and Néron [29, 30] that used the notion of *dual feasible functions*, inspired from bin-packing lower bounds, in order to derive destructive lower bounds.

The main feature of these bounds based on ILP-formulations is that the gap to optimality is often very small, but the computational time remains important, and then they cannot be used, for instance, in branch-and-bound methods.

2.4 Exact procedures

This section presents various exact methods, such as dynamic programming, and branch-and bound procedures which have been applied to solve the RCPSP to optimality.

2.4.1 Dynamic programming approach

Dynamic programming is an approach used to decompose problems into sub-problems and combine the solutions from each sub-problem into a complete solution for the original problem. A dynamic programming approach developed by Carruthers and Battersby [32] is the first effort to solve the RCPSP. Their objective is to find the expected maximum path length by reinterpreting the original problem as finding the maximum path length of the final activity of the network using the symmetry of the problem. Their approach made an advance in critical path methods, but it cannot handle practical networks.

2.4.2 Constraint programming

Another approach consists in modeling the problem as a set of variables (having defined domains) and constraints between these variables. Constraint programming consists then in finding a solution that satisfies all constraints.

We present two methods which fall in this category. The first one is the procedure proposed by Laborie [72], which uses the minimal forbidden set concept. Starting from a valid lower bound and applying a binary search, the procedure tries to find the best solution in the range [Lower Bound, Upper Bound]. The backtrack procedure of this method detects a minimal forbidden set and then *resolves* it. The resolution consists in the addition of a precedence constraint in order to break up the minimal forbidden set.

If the time limit is reached then the best lower bound is returned. To the best of our knowledge, this method appears to be the most efficient one for 60-activity instances of the PSPLIB benchmark (see Artigues et al. [5])
The second method was introduced by Liess and Michelon [75]. The basic idea consists in substituting the resource constraints by a set of *sub-constraints* generated as needed. Each of these sub-constraints corresponds to a set of tasks that cannot be executed together without violating one of the resource constraints. A filtering algorithm for these sub-constraints was developed.

2.4.3 Branch-and-Bound procedures

Branch-and-bound procedures build a search tree in order to explore implicitly the search space. At each node of the search tree (after computing lower bounds, upper bounds and possibly time-bound adjustments), the search space corresponding to the current node is partitioned into subsets such that the union of these subsets corresponds to the set of solutions of the current node. In the following, we present the *most* important branching schemes found in the literature.

A first natural way to review all possible combinations (that is used also in machine scheduling problems) consists in associating each node of the search tree with a partial feasible solution. The branching scheme consists of adding one *eligible* activity (whose predecessors are scheduled) to a partial schedule [90, 98]. Because of the cumulative nature of the RCPSP, feasible subsets of activities (instead of one activity) can be added to the partial solution [12]. This branching scheme will be more explained in Chapter 5.

Carlier and Latapie [27] proposed another branching scheme based on the reduction of time-windows of activities. The main idea of this branching scheme, is to determine the time-window of each activity. At each node, an activity is chosen and two nodes are created. In the first one the starting times of the chosen activity are restricted to the first half of the set of the feasible starting times, whereas in the second node they are restricted to the second half of this set. The main drawback of this approach is that the depth of the search tree depends on the time-windows of activities. Thus, in general, the search tree is very deep and the procedure falls to find optimal solutions.

Another branch-and-bound proposed by Brucker et al. [25] introduces disjunctive constraints between pairs of activities or places these activities in parallel. The branching starts from a graph representing a set of conjunctions, which means the precedence constraints, and disjunctions induced by the resource constraints. Their computation results based on the test data of Kolish et al. [66] showed that their algorithm performed well but needs to be improved.

The most efficient branch-and-bound algorithm was developed by Demeulemeester and Herroelen [40, 41]. The algorithm is based on a depth-first solution strategy in which nodes represent feasible partial solutions. At each node, if the eligible activities can be added then a node is created. Otherwise, if a resource conflict occurs, then the algorithm enumerates all possible combinations of activities that can be delayed in order to break up the conflict. The procedure enumerates only minimal (in the sense of union) combinations, i.e., a subset of a combination cannot be delayed in order to break up the conflict. Several dominance rules were also proposed by the authors in order to reduce the search space. Among all dominance rules, the Cutset dominance rule is undoubtedly the most efficient one. It consists at storing a list of non dominated partial schedules in a Hash table. This rule allowed to solve the 30-activity instances of the well-known PSPLIB benchmark [66] in a very short computation time. The drawback of this rule is that the list becomes very huge so the search in the table increases drastically and so the computation time. Its use is then limited to small or medium instances.

2.5 Heuristics

The $\mathcal{NP}$-hardness of the vast majority of scheduling problems incorporating realistic models and assumptions often necessitates the use of algorithms which run in reasonable time but do not guarantee an optimal solution. The term *heuristic* is commonly used to describe these algorithms.

2.5.1 Constructive heuristics

Constructive heuristics are simple algorithms allowing to quickly obtain a feasible solution. For the RCPSP, constructive heuristics consist of two main components: the *priority rule* and the *scheduling scheme*.
The priority rule is a kind of measure used for ordering activities in a priority list. The first activity in the list is the first activity to be scheduled, the second one is the second activity to be scheduled and so on. In order to respect the precedence constraints, it should be noted that each activity must be after its predecessors in the priority list.
The scheduling scheme determines how to choose the starting time for the activity to be scheduled. We distinguish two main scheduling schemes:

- The serial scheduling scheme [61] schedules the activities in their first possible starting time by respecting the precedence and resource constraints.
- The parallel scheduling scheme [21] looks at the completion times of the activities already scheduled. At each instant, this scheme selects the activities that can be scheduled (i.e. those whose predecessors are already scheduled) and schedules them at this time point provided that there is no resource conflict.

The survey of Kolisch and Hartmann [65] lists all scheduling schemes as well as priority rules found in the literature and gives a computational study.

2.5.2 Metaheuristics

Metaheuristics are improvement algorithms that start from one or more feasible solutions built from constructive heuristics or randomly. Several operations are performed on these solutions to build better ones. Operations in metaheuristics are generally based on the imitation of natural phenomena (e.g. genetic algorithms) or physical ones (e.g. simulated annealing) or on the study of the behavior of a group of individuals (ant colony, bee colony . . .). The main drawback of metaheuristics consists in the fact that convergence to the global optimum is not guaranteed: improvements can lead to a local optimum.

Several metaheuristics have been proposed for the RCPSP. As an example, we cite the very efficient genetic algorithms proposed by Hartmann [53, 55] and Valls et al. [105, 106], the simulated annealing procedure proposed by Bouleimen and Lecocq [20] (which appears to be one of the most efficient simulated annealing approaches) and the tabu search algorithms of Lee and Kim [74], Baar et al. [7] as well as Artigues et al. [6].

In recent years, the research in the field of metaheuristics for the RCPSP turned to hybrid approaches. Hybrid approaches combine two or three classical metaheuristic procedures in one algorithm in order to improve the results. Such algorithms include the procedures introduced by Kim et al. [62] and Agarwal et al. [1] which combines genetic algorithms with fuzzy logic approach. Other procedures include the algorithms of Wu et al. [111], Debels et al. [37] and Tseng and Chen [104]. One of the most recent articles in this area uses the shuffled frog-leaping algorithm, which combines the benefits of genetic-based memetic algorithm and the social behavior-based PSO algorithm [46].

2.6 Benchmark instances

One of the first set of instances has been proposed by Patterson [89]. It consists of a set of 110 instances. The size of these instances varies from 7 to 50 activities (22 activities on average) for a number of resources ranging from 1 to 3. Very good results were obtained for these instances in particular by the method proposed by Demeulemeester and Herroelen [40]. The second version of their method [41], improves greatly the results: all the 110 instances are solved in an average time of 0.025 seconds. At the late 90's, these instances became obsolete and does not represent a set of reference instances for the RCPSP anymore.

An alternative was proposed by Kolish et al. [66], through the provision of a generator of instances and an establishment of a library of instances. These instances are mainly characterized by three factors, which reflect the number of precedences, the average number of resources used by activities, and the average amount of resource used by each activity depending on the amount of available resource. Once again the method proposed by Demeulemeester and Herroelen [41] is extremely efficient since it solves all instances of 30 activities. However, the authors highlighted the limits of their approach that requires 500MB (essential for the Cut-Set dominance rule) of memory for processing instances with 60 activities.

Baptiste and Le Pape [11] have defined a new parameter, called *ratio of disjunctions*, which expresses the percentage of pairs of activities which can be performed simultaneously (two activities can be executed simultaneously if there is no precedence relationship between them and the sum of their demands on a resource does not exceed the capacity thereof) relative to the total number of pairs of activities. Using this criterion, the authors showed that the three main factors for generating instances of Kolisch et al. do not fully reflect the cumulative nature of the problem. The authors have developed a set of 39 instances of reasonable size (20 and 25 tasks), known as highly cumulative:

the disjunction ratio is on average equal to 0.33 for those instances when it is equal to 0.53 for the most cumulative instances of Kolisch et al. [66]. Baptiste and Le Pape [11] showed that, on the traditional instances (Patterson [89], Kolisch et al. [66]), the use of tools such as energetic reasoning show a poor efficiency. Instead, these tools are indispensable to solve effectively strongly cumulative problems.

Also, we cite **PACK** instances proposed by Carlier and Néron [28, 29]. The authors generated a total of 55 instances which include two types of instances. The first type have a strong disjunctive component. The second type contains activities which consumption may not exceed half the capacity of the resource in order to obtain highly cumulative instances. The authors provide also effective lower bounds for this benchmark.

For the numerical tests conducted throughout the thesis, we focus on **KSD** instances of Kolisch et al. In the following, we detail a little more the composition of this benchmark instances called **PSPLIB**. The test instances were generated according to the following parameters:

- The execution times p_j of the activities are generated between 1 and 10.
- Each activity has three successors at most.
- Each instance has 4 resources.
- The sizes of the instances are equal to 30, 60, 90 and 120 activities.
- The precedence graph is generated according to the parameter **Network Complexity (NC)** that belongs to the set {1.5; 1.8; 2.1}.
- The resources are generated according to two parameters:
  - **Resource Factor (RF)** which determines the consumption of each activity. RF belongs to {0.25; 0.5; 0.75; 1}.
  - **Resource Strength (RS)** which determines the capacity of a resource. RS belongs to {0.2; 0.5; 0.7; 1} for instances of 30, 60 et 90 activities, and belongs to {0.1; 0.2; 0.3; 0.4; 0.5} for instances of 120 activities.

For each triplet (NC, RF, RS), 10 instances are generated for a total of 480 instances for sizes 30, 60 and 90 activities, and 600 instances for the size 120 activities which gives a total of 2040 instances.

2.7 Conclusion

In this chapter, we gave literature review of the methods to solve the resource constrained project scheduling problem. We can divide these methods into two categories: exact and heuristic methods. The first category consists in solving exactly the problem and includes various techniques such as lower bounding approaches, mathematical formulations, constraint programming, etc. These methods are efficient for small instances but fall to solve medium sized instances. An alternative to the exact methods, heuristic approaches try to find good solutions (not necessary optimal) in reasonable amount of time. Thus, heuristics methods can handle large sized instances.

In this thesis, we propose new approaches to solve exactly the problem. The first part (chapter 3 and 4) is dedicated to lower bounding approaches. In the next chapter, we begin our study by proposing various enhancements of the energetic reasoning in order to derive new and efficient lower bounds.

Chapter 3

Enhanced Energetic Reasoning-Based Lower Bounds for the Resource Constrained Project Scheduling Problem

3.1 Introduction

Actually, it is widely recognized that the effectiveness of exact enumerative algorithms strongly relies on the performance of the embedded lower bounds. Indeed, the lower bounding procedure that is invoked within the enumerative algorithm should ideally be both *effective* and *efficient* (that is, it should yield a tight lower bound while requiring a short computing time). However, a review of the lower bounds that have been proposed for the RCPSP (see Section 2.3) reveals that these bounds includes fast lower bounds that exhibit an erratic (i.e., nonrobust) behavior and tight lower bounds that require a substantial computing burden. Clearly, none of these two categories is fully satisfactory with regard to the effective solution of large-scale RCPSPs. The objective of this chapter is to propose new lower bounds that would prove both tight and fast. More precisely, we make the following contributions:

1) We introduce the concept of *reduced instance* and show how it could be used to derive a new enhanced lower bound using previously proposed lower bounds.

2) We propose new lower bounds that are derived through enhancing the *Energetic Reasoning* (ER) (Lahrichi [73]). These enhancements are achieved through nontrivial generalizations of the so-called *Revisited Energetic Reasoning* (RER) that was initially developed by Hidri et al. [58] in the context of multiprocessor scheduling.

3) We introduce a further improvement of ER using *Dual Feasible Functions* (DFFs).

4) We present the results of a comprehensive computational study that provides evidence that the best proposed lower bound exhibits an excellent performance while requiring a modest computing time.

The remainder of this chapter is organized as follows. In Section 3.2, we begin by reviewing the classical energetic reasoning. In Section 3.3, we show how the definition of a reduced auxiliary RCPSP instance offers an interesting way for deriving new effective lower bounds. In Section 3.4, we present several generalizations of the Revisited Energetic Reasoning. In Section 3.5, we explain how these bounds might be subsequently improved through using DFFs. The results of a comprehensive computational study of the proposed bounds are presented in Section 3.6.

It is worth noting that most of the material presented in this chapter has been published in Kooli et al. [69, 68] and Haouari et al. [51].

3.2 Classical energetic reasoning

In this chapter, the energetic reasoning is a central issue. Lower bounds that we propose are based on this approach, using for instance the concept of mandatory part of an activity in a given time-interval. All these bounds belong to the class of *destructive lower bounds*. At this point, it is worth recalling that a destructive bound could be derived using the following general approach. Starting from a trial value C, feasibility tests are carried out to detect an infeasibility (that is, the makespan cannot be shorter than C). If an infeasibility is detected, then $C+1$ is a valid lower bound (or, alternatively a dichotomous search might be used). The process is reiterated until no infeasibility is detected. In this section, we briefly review both feasibility tests and time-bound adjustments of classical energetic reasoning, initially discussed by Lahrichi [73], Erschler et al. [45] and Baptiste et al. [12].

Let $[t_1, t_2]$ be a time-interval. The work of an activity j $(j \in \mathcal{A})$ in this time-interval is given by the minimum between its left work denoted $W^l_{jk}(t_1, t_2)$, i.e., when the activity starts at its release date, and its right work denoted as $W^r_{jk}(t_1, t_2)$, i.e., when the activity ends at its deadline.

$$\begin{aligned}
W^l_{jk}(t_1, t_2) &= b_{jk} \min(t_2 - t_1, p_j, \max(0, r_j + p_j - t_1)) \\
W^r_{jk}(t_1, t_2) &= b_{jk} \min(t_2 - t_1, p_j, \max(0, t_2 - d_j + p_j)) \\
W_{jk}(t_1, t_2) &= \min(W^l_{jk}(t_1, t_2), W^r_{jk}(t_1, t_2))
\end{aligned}$$

The total work $W_k(t_1, t_2)$ over the time-interval $[t_1, t_2]$ and resource k is defined as follows:

$$W_k(t_1, t_2) = \sum_{j \in \mathcal{A}} W_{jk}(t_1, t_2)$$

Property 1. *If there exists a resource $k \in \mathcal{R}$ such that $W_k(t_1, t_2) > B_k(t_2 - t_1)$ then the instance is infeasible. Consequently, $C+1$ is a valid lower bound on the project duration.*

If no contradiction (i.e., infeasibility) is detected, then a time-bound adjustment may be considered (Baptiste et al. [12]) using the concept of slack of one activity $j \in \mathcal{A}$ for a given resource $k \in \mathcal{R}$ on a given time-interval $[t_1, t_2]$:

$$s_{jk}(t_1, t_2) = B_k(t_2 - t_1) - \{W_k(t_1, t_2) - W_{jk}(t_1, t_2)\} \tag{3.1}$$

Property 2. *(Adjustment procedure)*

$$\textit{if } s_{jk}(t_1, t_2) < W^l_{jk}(t_1, t_2) \textit{ then } r_j \leftarrow \max(r_j, t_2 - \left\lfloor \frac{s_{jk}(t_1, t_2)}{b_{jk}} \right\rfloor)$$
$$\textit{if } s_{jk}(t_1, t_2) < W^r_{jk}(t_1, t_2) \textit{ then } d_j \leftarrow \min(d_j, t_1 + \left\lfloor \frac{s_{jk}(t_1, t_2)}{b_{jk}} \right\rfloor)$$

These time-bound adjustments are propagated on the precedence graph G.

Notice that an important issue is related to the determination of the relevant set of time-intervals on which it may be useful to both check feasibility conditions and time-bound adjustments. Baptiste et al. [12] have proved that there is a quadratic number of such time-intervals.

A pseudo-code for the computation of a lower bound based on the classical energetic reasoning is described in Algorithm 1.

Algorithm 1 Energetic Reasoning Procedure

```
1:  $T_1$: Table containing the $t_1$ values of the time-intervals $[t_1, t_2]$
2:  $T_2$: Table containing the $t_2$ values of the time-intervals $[t_1, t_2]$
3:  $C \leftarrow$ Capacity Bound
4:  repeat
5:    ComputeTimeIntervals($T_1$,$T_2$)
6:    feasible $\leftarrow$ true
7:    j $\leftarrow$ 0
8:    while (feasible=true) and ($j \leq T_1.size$) do
9:      $i \leftarrow T_2.size$
10:     while (feasible=true) and ($T_1[j] \leq T_2[i]$) do
11:       feasible=FeasibilityTest($T_1[j]$,$T_2[i]$)
12:       $i \leftarrow i - 1$
13:     end while
14:     $j \leftarrow j + 1$
15:   end while
16:   if feasible=false then
17:     $C \leftarrow C + 1$
18:   end if
19: until feasible=true
```

3.3 Improved energetic reasoning

Given a trial value C and a time-interval $[t_1, t_2]$, we define:

$$p_j(t_1,t_2) = \min(t_2 - t_1, p_j, \max(0, r_j + p_j - t_1), \max(0, t_2 - d_j + p_j))$$

a lower bound on the total processing time of activity $j \in \mathcal{A}$ during $[t_1, t_2]$.

Define $\mathcal{A}(t_1,t_2) = \{j \in \mathcal{A} : (r_j + p_j > t_1) \wedge (d_j - p_j < t_2)\}$ the set of activities that are necessarily processed during $[t_1, t_2]$.
Given an RCPSP instance, a trial value C, and a time-interval $[t_1, t_2]$ we construct an associated *reduced instance* in the following way:

- the set of activities is $\mathcal{A}(t_1,t_2)$,
- the processing time of activity $j \in \mathcal{A}(t_1,t_2)$ is $p_j(t_1,t_2)$,
- the values of the availabilities of the renewable resources, the resources requirements as well as the precedence constraints are similar to those of the genuine instance.

Proposition 1. *If the optimal makespan of the reduced instance is strictly larger than the time-window width ($t_2 - t_1$) then $C + 1$ is a valid lower bound on the optimal makespan of the genuine instance.*

Proof. It suffices to observe that if the optimal makespan of the reduced instance is strictly larger than the time-window width $(t_2 - t_1)$ then the mandatory parts of the activities that belong to $\mathcal{A}(t_1,t_2)$ cannot be completed within $[t_1, t_2]$. Thus, the genuine instance is infeasible. □

In our implementation, we invoke a general-purpose MIP solver to solve an exact feasibility problem through using the following time-indexed formulation (Christofides et al. [35]).

$$\text{Find } \delta \tag{3.2}$$

s.t.

$$\sum_{t=\max(r_j,t_1)}^{d_j-p_j} \delta_{jt} = 1, \qquad \forall j \in \mathcal{A}(t_1,t_2), \tag{3.3}$$

$$\sum_{\tau=t}^{d_i-p_i} \delta_{i\tau} + \sum_{\tau=\max(r_j,t_1)}^{t+p_i-1} \delta_{j\tau} \leq 1,\ \forall (i,j) \in A,\ t \in [r_j - p_i + 1, d_i - p_i] \tag{3.4}$$

$$\sum_{j \in \mathcal{A}(t_1,t_2)} b_{jk} \sum_{s=\max(r_j,t-p_j+1)}^{\min(t,d_j-p_j)} \delta_{js} \leq B_k, \qquad \forall k \in \mathcal{R},\ t \in [t_1,t_2], \tag{3.5}$$

$$\delta_{jt} \in \{0,1\}, \qquad \forall j \in \mathcal{A}(t_1,t_2),\ t \in [\max(r_j,t_1), d_j - p_j], \tag{3.6}$$

where $\delta_{jt} = 1$, if activity j starts at time t, and 0 otherwise, $\forall j \in \mathcal{A}(t_1,t_2)$, $t \in [\max(r_j,t_1), d_j - p_j]$. Clearly, Constraint (3.3) requires that each activity has exactly one starting time. Constraints (3.4) and (3.5) enforce the precedence and resource constraints, respectively.

In the sequel, we denote IER^* the bound based on Proposition 1. In our implementation, and for the sake of efficiency, only reduced instances whose number of activities is smaller than 12 and having a time-window width shorter than 25 are considered.

A straightforward (and practically useful) consequence of Proposition 1 is the following:

Corollary 1. *If a lower bound on the optimal makespan of the reduced instance is strictly larger than the time-window width $(t_2 - t_1)$ then $C + 1$ is a valid lower bound on the optimal makespan of the genuine instance.*

In the sequel, we refer to *improved energetic reasoning* the variant of the energetic reasoning that is based on Proposition 1 or Corollary 1. Within this context, it is easy to realize that if the capacity bound is used then the improved energetic reasoning amounts to the classical energetic reasoning. In our experiments, we implemented the improved energetic reasoning together with the critical sequence as well as the m-machine lower bounds.

Example 2. *Consider a project with 5 activities, represented by Figure 3.1 and a single resource of capacity $B_1 = 11$.*

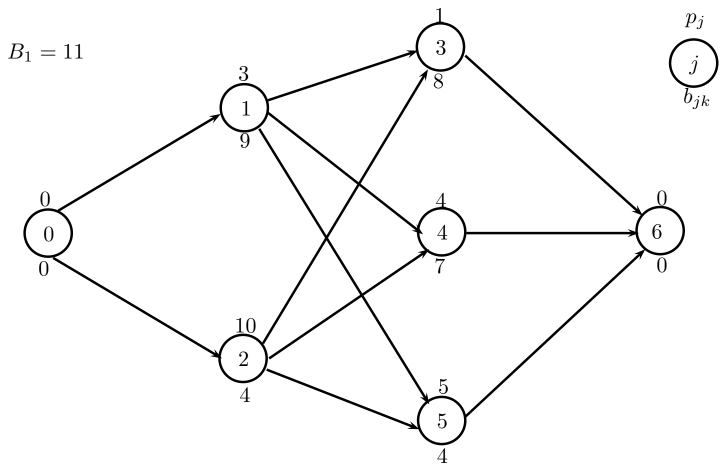


Figure 3.1: Improved energetic reasoning project example

If we apply Algorithm 1, we obtain a lower bound equal to 14.

Suppose that $t_1 \in \{r_j, d_j - p_j; j \in \mathcal{A}\}$ and $t_2 \in \{d_j, r_j + p_j; j \in \mathcal{A}\}$,which gives (with $C = 14$):

- $t_1 \in \{0, 7, 10, 12, 13, 14\}$
- $t_2 \in \{0, 3, 10, 11, 12, 14\}$

Now, we add to the feasibility tests, the calculation of parallel machines based lower bound. We take $m = 1$, as a number of machines. Thus, we build $1|r_j, q_j|C_{max}$ problems. It is worth noting that the m-machine lower bound LB_P on the original instance is equal to 8. At time-interval $[7, 14]$, the processing times of the activities are:

j	1	2	3	4	5
$p_j(7, 14)$	0	3	1	4	2

The reduced instance is then constituted by activities: 2, 3, 4 and 5 (see Figure 3.2).

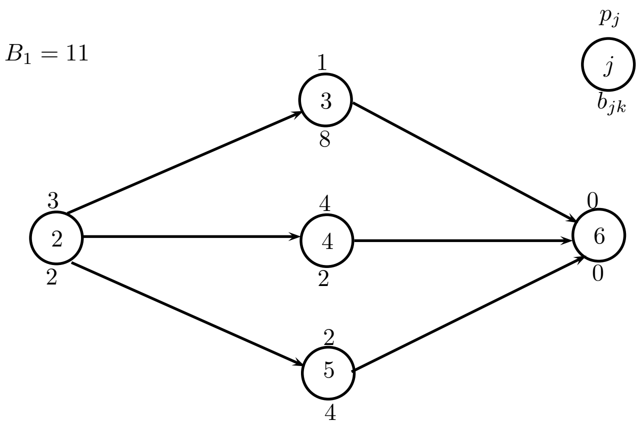


Figure 3.2: Reduced instance on time-interval [7,14]

The calculation of the release dates and due dates on this reduced instance yields:

- $r_2 = 7$; $r_3 = r_4 = r_5 = 10$
- $d_2 = 10$; $d_3 = d_4 = d_5 = 14$

Since m is equal to 1, then $u_1^m = 6$. Only activities 3 and 4 have resource requirements higher or equal than 6. Thus, we get a $1|r_j, q_j|C_{max}$ problem constituted by two jobs with the following data:

j	p_j	r_j	q_j
1	1	10	0
2	4	10	0

The computation of the m-machine lower bound on this reduced instance gives the value $LB_P = 15$ which is higher than 14. So, we detect an infeasibility. Finally, we find that the improved energetic reasoning-based lower bound IER is equal to 15.

3.4 Revisited energetic reasoning

In this section, we introduce new bounds for the RCPSP that are based on the so-called *Revisited Energetic Reasoning* (RER). The RER was initially introduced by Hidri et al. [58] in the context of parallel machine scheduling with heads and tails. It aims at strengthening the classical ER by improving the computation of the work through formulating an integer programming (IP) problem. In the following, we present an example that introduce the basic idea of the RER. Then, some additional notation and several variants of RER-based lower bounds are presented.

3.4.1 Revisited energetic reasoning principle

We consider a RCPSP instance with the following data : $n = 6$, $K = 1$ and $B_1 = 2$. For each activity j, we have $b_{jk} = 1$, $r_j = 1$, $p_j = 3$ and $d_j = 8$; $j = 1, \ldots, 6$. The time-interval $[t_1, t_2]$ is [2,6].

If we apply the classical energetic reasoning then we get: $W^l_{j1}(2,6) = 2$ and $W^r_{j1}(2,6) = 1$, $j = 1, \ldots, 6$. Thus, $W_{j1}(2,6) = 1$; $j = 1, \ldots, 6$. The total work over the time-interval [2,6] is then $W_1(2,6) = 6$ and no infeasibility is detected.

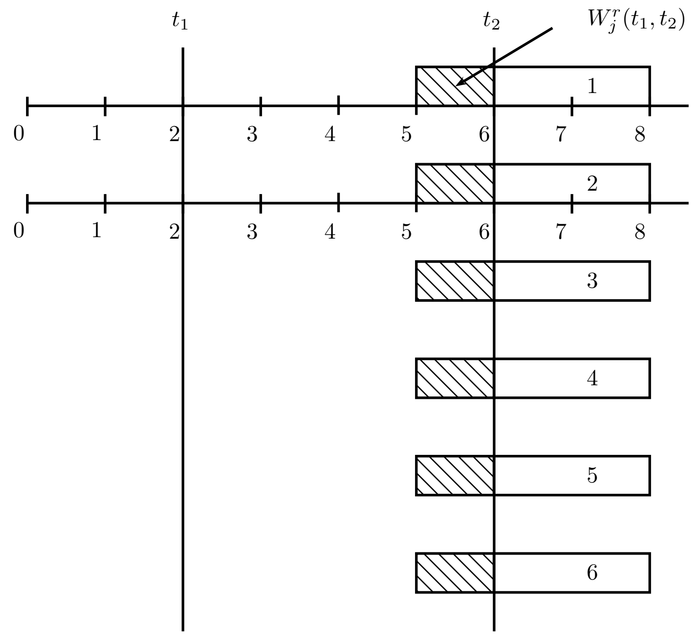


Figure 3.3: Classical energetic reasoning principle

Otherwise, we remark that to obtain $W_1(2,6)$ all the activities finish up processing at $d_j = 8$ (see Figure 3.3). Thus, at $t_2 = 6$, we have 6 activities scheduled on a resource with capacity $B_1 = 2$. The basic idea behind the RER is that in any feasible schedule at most 2 activities can be processed on this resource at the same time. As a consequence, we can have at most 2 activities at t_1, 2 activities at t_2 and the remaining activities are in the inside (see Figure 3.4).

Thus, for this example, we have:

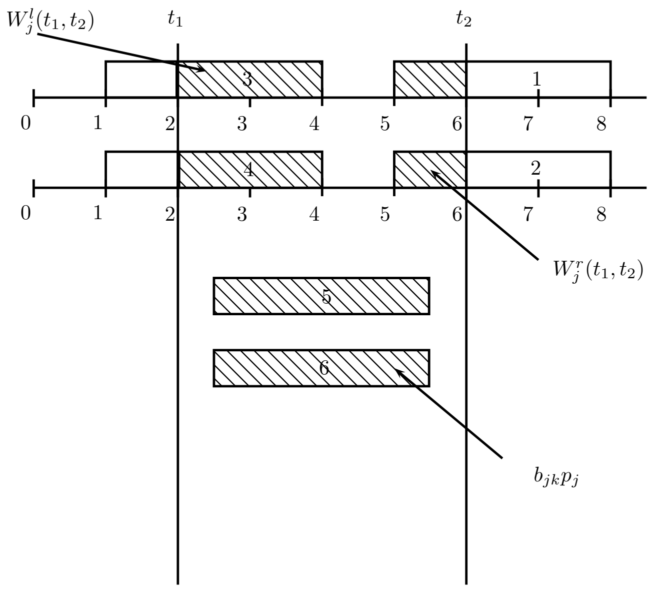


Figure 3.4: Revisited energetic reasoning principle

$$\begin{array}{ll} W_{11}(2,6) = W_{21}(2,6) = 1 & \\ W_{31}(2,6) = W_{41}(2,6) = 2 & \Rightarrow W_1(2,6) = 1 \cdot 2 + 2 \cdot 2 + 3 \cdot 2 = 12 > 8\,(= 4 \cdot 2) \\ W_{51}(2,6) = W_{61}(2,6) = 3 & \end{array}$$

An infeasibility is then detected.

3.4.2 Notation

Define $\mathcal{A}_0(t_1,t_2) = \{j \in \mathcal{A}(t_1,t_2) : (r_j + p_j \geq t_2) \wedge (d_j - p_j \leq t_1)\}$. Each activity $j \in \mathcal{A}_0(t_1,t_2)$ must be scheduled over the whole interval $[t_1,t_2]$. Thus, we can then remove activity j from $\mathcal{A}(t_1,t_2)$ and decrease the capacity B_k of resource k by b_{jk} units. Hence, the actual resource capacity $m_k(t_1,t_2)$ is computed as follows:

$$m_k(t_1,t_2) \leftarrow B_k - \sum_{j \in \mathcal{A}_0(t_1,t_2)} b_{jk}$$

In the sequel, we shall say that an activity j is:

- placed at the *left position* if it is scheduled at its release date
- placed at the *right position* if it finishes processing at its due date
- placed at the *inside position* if it is entirely processed within the time-interval.

The set $\mathcal{A}(t_1,t_2) \setminus \mathcal{A}_0(t_1,t_2)$ is partitioned into the following seven subsets:

- $\mathcal{A}_L = \{j \in \mathcal{A}(t_1,t_2) \setminus \mathcal{A}_0(t_1,t_2) : (r_j + p_j < t_2) \wedge (d_j - p_j \leq t_1)\}$. The activities of this subset must be scheduled at the left position.
- $\mathcal{A}_R = \{j \in \mathcal{A}(t_1,t_2) \setminus \mathcal{A}_0(t_1,t_2) : (r_j + p_j \geq t_2) \wedge (d_j - p_j > t_1)\}$. The activities of this subset must be scheduled at the right position.
- $\mathcal{A}_I = \{j \in \mathcal{A}(t_1,t_2) \setminus \mathcal{A}_0(t_1,t_2) : (t_1 < r_j) \wedge (d_j < t_2)\}$. The activities of this subset must be scheduled inside the time-interval.
- $\mathcal{A}_{LI} = \{j \in \mathcal{A}(t_1,t_2) \setminus \mathcal{A}_0(t_1,t_2) : (r_j \leq t_1) \wedge (d_j < t_2) \wedge (d_j - p_j > t_1)\}$. The activities of this subset are either scheduled at the left position or inside the time-interval.
- $\mathcal{A}_{RI} = \{j \in \mathcal{A}(t_1,t_2) \setminus \mathcal{A}_0(t_1,t_2) : (r_j > t_1) \wedge (d_j \geq t_2) \wedge (r_j + p_j < t_2)\}$. The activities of this subset are either scheduled at the right position or inside the time-interval.
- $\mathcal{A}_{LR} = \{j \in \mathcal{A}(t_1,t_2) \setminus \mathcal{A}_0(t_1,t_2) : (r_j \leq t_1) \wedge (d_j \geq t_2) \wedge (r_j + p_j < t_2) \wedge (d_j - p_j > t_1) \wedge (p_j \geq t_2 - t_1 - 1)\}$. The activities of this subset are either scheduled at the left position or at the right position (since they cannot be inside the time-interval.)
- $\mathcal{A}_{LIR} = \{j \in \mathcal{A}(t_1,t_2) \setminus \mathcal{A}_0(t_1,t_2) : (r_j \leq t_1) \wedge (d_j \geq t_2) \wedge (r_j + p_j < t_2) \wedge (d_j - p_j > t_1) \wedge (p_j < t_2 - t_1 - 1)\}$. The activities of this subset can be at any position within the time-interval.
- $\mathcal{A}^f = \{j \in \mathcal{A}(t_1,t_2) \setminus \mathcal{A}_0(t_1,t_2) : (r_j + p_j) > (t_2 + t_1)/2 \wedge (d_j - p_j) \leq (t_2 + t_1)/2\}$. The activities of this subset are necessary in execution at the middle of the time-interval.

We also denote by:

$\mathcal{A}_{\mathcal{L}} = \mathcal{A}_{LI} \cup \mathcal{A}_{LR} \cup \mathcal{A}_{LIR}$; the set of activities that may be placed at the left position.
$\mathcal{A}_{\mathcal{R}} = \mathcal{A}_{RI} \cup \mathcal{A}_{LR} \cup \mathcal{A}_{LIR}$; the set of activities that may be placed at the right position.
$\mathcal{A}_{\mathcal{I}} = \mathcal{A}_{LI} \cup \mathcal{A}_{RI} \cup \mathcal{A}_{LIR}$; the set of activities that may be placed inside the interval.
$\mathcal{A}^f_{\mathcal{L}} = \{j \in \mathcal{A}_{\mathcal{L}} : r_j + p_j > (t_2 + t_1)/2\}$; the set of activities that may be in execution in the middle of the interval if they are scheduled at the left position.
$\mathcal{A}^f_{\mathcal{R}} = \{j \in \mathcal{A}_{\mathcal{R}} : d_j - p_j \leq (t_2 + t_1)/2\}$; the set of activities that may be in execution in the middle of the interval if they are scheduled at the right position.
$\mathcal{A}^f_{\mathcal{I}} = \{j \in \mathcal{A}_{\mathcal{I}} : \max(r_j, t_1) + p_j > (t_2 + t_1)/2 \wedge \min(d_j, t_2) \leq (t_2 + t_1)/2\}$; the set of activities that may be in execution in the middle of the interval if they are scheduled in the inside.

In the sequel, and for the sake of clarity, we shall replace $\mathcal{A}(t_1,t_2) \backslash \mathcal{A}_0(t_1,t_2), m_k(t_1,t_2)$, $W^l_{jk}(t_1,t_2)$, $W^r_{jk}(t_1,t_2)$ and $W_k(t_1,t_2)$ by $\mathcal{A}$, m_k, W^l_{jk}, W^r_{jk} and W_k, respectively.

3.4.3 Simple feasibility conditions

Hidri et al. [58] derived several *simple feasibility conditions* that are valid for the identical parallel machine problem with release dates and delivery times. These conditions can be easily adapted to the RCPSP. These feasibility conditions are the following:

Condition 1: If there exists a resource $k \in \mathcal{R}$ and an activity $j \in \mathcal{A}$ such that $b_{jk} > m_k$ then the instance is infeasible.

Condition 2: If there exists a resource $k \in \mathcal{R}$ such that $m_k < 0$ then the instance is infeasible.

Condition 3: If there exists a resource $k \in \mathcal{R}$ such that $m_k = 0$ and an activity $j \in \mathcal{A}$ such that $b_{jk} > 0$ then the instance is infeasible.

Condition 4: If there exists a resource $k \in \mathcal{R}$ and an activity $j \in \mathcal{A}_{LR}$ such that $b_{jk} > m_k - \min(\sum\limits_{j \in \mathcal{A}_L} b_{jk}, \sum\limits_{j \in \mathcal{A}_R} b_{jk})$ then the instance is infeasible.

Condition 5: If there exists a resource $k \in \mathcal{R}$ such that $\max(\sum\limits_{j \in \mathcal{A}_L} b_{jk}, \sum\limits_{j \in \mathcal{A}_R} b_{jk}) > m_k$ then the instance is infeasible.

Condition 6: If there exists a resource $k \in \mathcal{R}$ such that $\sum\limits_{j \in \mathcal{A}_L \cup \mathcal{A}_R \cup \mathcal{A}_{LR}} b_{jk} > 2m_k$ then the instance is infeasible.

Condition 7: If there exists a resource $k \in \mathcal{R}$ such that $\sum\limits_{j \in \mathcal{A}^f} b_{jk} > m_k$ then the instance is infeasible.

Moreover, let:

$$\alpha = \min_{j \in \mathcal{A}_L}(r_j + p_j)$$

$$\beta = \max_{j \in \mathcal{A}_R}(d_j - p_j)$$

$$\gamma = \min_{j \in \mathcal{A}_{LR}}(r_j + p_j)$$

$$\delta = \max_{j \in \mathcal{A}_{LR}}(d_j - p_j)$$

With these new definitions, we have the following feasibility tests:

Condition 8: If there exists a resource $k \in \mathcal{R}$ such that $\sum\limits_{j \in \mathcal{A}_L} b_{jk} = m_k$ and $\min\limits_{j \in \mathcal{A}_I \cup \mathcal{A}_{LI}} (d_j - p_j) < \alpha$ then the instance is infeasible.

Condition 9: If there exists a resource $k \in \mathcal{R}$ such that $\sum\limits_{j \in \mathcal{A}_R} b_{jk} = m_k$ and $\min\limits_{j \in \mathcal{A}_I \cup J_{RI}} (r_j + p_j) < \beta$ then the instance is infeasible.

Condition 10: If there exists a resource $k \in \mathcal{R}$ such that $\sum\limits_{j \in J_L \cup J_R \cup J_{LR}} b_{jk} = 2m_k$ and $\min\limits_{j \in \mathcal{A}_I \cup \mathcal{A}_{LI} \cup \mathcal{A}_{RI} \cup \mathcal{A}_{LIR}} p_j > \max(\beta, \delta) - \min(\alpha, \gamma)$ then the instance is infeasible.

In addition to that, we have the following adjustments rules:

Rule 1: If $\sum_{j \in \mathcal{A}_L} b_{jk} = m_k$ then $r_j = \max(r_j, \alpha)$; $\forall j \in \mathcal{A} \backslash \mathcal{A}_L$

Rule 2: If $\sum_{j \in \mathcal{A}_R} b_{jk} = m_k$ then $d_j = \min(d_j, \beta)$; $\forall j \in \mathcal{A} \backslash \mathcal{A}_R$

Rule 3: If $\sum_{j \in \mathcal{A}_L \cup \mathcal{A}_R \cup \mathcal{A}_{LR}} b_{jk} = 2m_k$ then $r_j = \max(r_j, \min(\alpha, \gamma))$; $\forall j \in \mathcal{A} \backslash (\mathcal{A}_L \cup \mathcal{A}_{LR})$ and $d_j = \min(d_j, \min(\beta, \delta))$; $\forall j \in \mathcal{A} \backslash (\mathcal{A}_R \cup \mathcal{A}_{LR})$

Example 3. *We resume Example 2. For the time-interval [0,10], we get the same reduced instance shown in Figure 3.2. Activity 2 has a processing part equal to the width of the interval. So, we remove it from the set $\mathcal{A}$ and we decrement the capacity of the resource. We, therefore, obtain a residual capacity equal to $m_1 = 11 - 4 = 7$. According to condition 1, we have $b_{11} = 9 > 7$ implying infeasibility. The classical energetic reasoning-based lower bound in addition to the simple feasibility conditions is equal to 17.*

3.4.4 Exact revisited energetic reasoning

For each resource k ($k \in \mathcal{R}$), we compute the residual capacities m_k^l, m_k^r and m_k^f over the time-interval $[t_1, t_2]$ available for activities of the set $\mathcal{A}_\mathcal{L}$, $\mathcal{A}_\mathcal{R}$ and $\mathcal{A} \backslash \mathcal{A}^f$, respectively, i.e.,

$$m_k^l = \min\Big(\sum_{j \in \mathcal{A}_\mathcal{L}} b_{jk}, m_k - \sum_{j \in \mathcal{A}_L} b_{jk}\Big)$$
$$m_k^r = \min\Big(\sum_{j \in \mathcal{A}_\mathcal{R}} b_{jk}, m_k - \sum_{j \in \mathcal{A}_R} b_{jk}\Big)$$
$$m_k^f = m_k - \sum_{j \in \mathcal{A}^f} b_{jk}$$

Now, we present a 0-1 programming model that computes an enhanced estimate of the total work over the time-interval $[t_1, t_2]$ for resource k. Toward this end, we define for each activity $j \in \mathcal{A}$ the following binary variables:

- $x_j = 1$ if activity j is scheduled at the left position, and 0 otherwise
- $y_j = 1$ if activity j is scheduled at the right position, and 0 otherwise
- $z_j = 1$ if activity j is scheduled inside the interval, and 0 otherwise

The model is the following:

$$
(P1)\quad
\begin{aligned}
&\overline{E}_k = \min \sum_{j\in\mathcal{A}_\mathcal{L}} W^l_{jk}x_j + \sum_{j\in\mathcal{A}_\mathcal{R}} W^r_{jk}y_j + \sum_{j\in\mathcal{A}_\mathcal{I}} W^i_{jk}z_j && (3.7)\\
&\text{s.t.}\\
&x_j + z_j = 1, \quad \forall j \in \mathcal{A}_{LI} && (3.8)\\
&y_j + z_j = 1, \quad \forall j \in \mathcal{A}_{RI} && (3.9)\\
&x_j + y_j = 1, \quad \forall j \in \mathcal{A}_{LR} && (3.10)\\
&x_j + y_j + z_j = 1, \quad \forall j \in \mathcal{A}_{LIR} && (3.11)\\
&\sum_{j\in\mathcal{A}_\mathcal{L}} b_{jk}x_j \le m^l_k && (3.12)\\
&\sum_{j\in\mathcal{A}_\mathcal{R}} b_{jk}y_j \le m^r_k && (3.13)\\
&\sum_{j\in J^f_\mathcal{L}} b_{jk}\,x_j + \sum_{j\in J^f_\mathcal{R}} b_{jk}\,y_j + \sum_{j\in J^f_\mathcal{I}} b_{jk}\,z_j \le m^f_k && (3.14)\\
&x_j, y_j, z_j \in \{0,1\}, \quad \forall j \in \mathcal{A}_{LI} \cup \mathcal{A}_{RI} \cup \mathcal{A}_{LR} \cup \mathcal{A}_{LIR}, && (3.15)
\end{aligned}
$$

where $W^i_{jk} \equiv b_{jk}p_j$.

Constraints (3.8)-(3.11) require that each activity is scheduled in exactly one position (left, right or inside). Constraints (3.12)-(3.13) enforce the left and right capacities of resource k, respectively. Constraint (3.14) enforce the resource constraint at the middle of the time-interval. Constraint (3.15) states that the variables are binary-valued.

The *improved work* $\overline{W}_k(t_1,t_2)$ is given by:

$$\overline{W}_k(t_1,t_2) = \overline{E}_k + \sum_{j\in\mathcal{A}_L} W^l_{jk} + \sum_{j\in\mathcal{A}_I} W^i_{jk} + \sum_{j\in\mathcal{A}_R} W^r_{jk}$$

We have the following result:

Property 3. *If there exists a resource $k \in \mathcal{R}$ such that $\overline{W}_k(t_1,t_2) > m_k(t_2-t_1)$ then the instance is infeasible.*

3.4.5 Relaxed revisited energetic reasoning

Since the solution of Model (3.7)-(3.15) might require an excessive computing time, it might be useful to consider relaxations of this model that are efficiently solvable. To that aim, we propose a relaxation that consists in decomposing the problem into two independent knapsack problems. This relaxation, hereafter called *relaxed revisited energetic reasoning* is described as follows.

First, we replace variables z_j of formulation $(P1)$ by:

$$z_j = 1 - x_j, \quad \forall j \in \mathcal{A}_{LI}$$
$$z_j = 1 - y_j, \quad \forall j \in \mathcal{A}_{RI}$$
$$z_j = 1 - x_j - y_j, \quad \forall j \in \mathcal{A}_{LIR}$$

Thus, (3.7) becomes:

$$\min \sum_{j \in \mathcal{A}_{LI} \cup \mathcal{A}_{RI} \cup \mathcal{A}_{LR} \cup \mathcal{A}_{LIR}} W^i_{jk} + \sum_{j \in \mathcal{A}_{\mathcal{L}}} (W^l_{jk} - W^i_{jk}) x_j + \sum_{j \in \mathcal{A}_{\mathcal{R}}} (W^r_{jk} - W^i_{jk}) y_j$$

Let:

$$\tilde{W}^l_{jk} = W^i_{jk} - W^l_{jk}, \quad \forall j \in \mathcal{A}_{\mathcal{L}}$$
$$\tilde{W}^r_{jk} = W^i_{jk} - W^r_{jk}, \quad \forall j \in \mathcal{A}_{\mathcal{R}}$$

Formulation $(P1)$ is equivalent to:

$$\text{(P2)} \quad
\begin{aligned}
& \max \sum_{j \in \mathcal{A}_{\mathcal{L}}} \tilde{W}^l_{jk} x_j + \sum_{j \in \mathcal{A}_{\mathcal{R}}} \tilde{W}^r_{jk} y_j && (3.16)\\
& \text{s.t.} \\
& x_j + y_j = 1, \quad \forall j \in \mathcal{A}_{LR} && (3.17)\\
& x_j + y_j \leq 1, \quad \forall j \in \mathcal{A}_{LIR} && (3.18)\\
& \sum_{j \in \mathcal{A}_{\mathcal{L}}} b_{jk} x_j \leq m^l_k && (3.19)\\
& \sum_{j \in \mathcal{A}_{\mathcal{R}}} b_{jk} y_j \leq m^r_k && (3.20)\\
& \sum_{j \in J^f_{\mathcal{L}}} b_{jk}\, x_j + \sum_{j \in J^f_{\mathcal{R}}} b_{jk}\, y_j + \sum_{j \in J^f_{\mathcal{I}}} b_{jk}\, z_j \leq m^f_k && (3.21)\\
& x_j, y_j \in \{0, 1\}, \quad \forall j \in \mathcal{A}_{LI} \cup \mathcal{A}_{RI} \cup \mathcal{A}_{LR} \cup \mathcal{A}_{LIR} && (3.22)
\end{aligned}$$

If we relax constraints (3.17), (3.18) and (3.21) then we obtain two independent knapsack problems. We solve exactly each one separately using dynamic programming. Let $\tilde{E}_k$ be the sum of the objective functions of the two problems, then:

$$\tilde{W}_k(t_1, t_2) = \sum_{j \in \mathcal{A}_{LI} \cup \mathcal{A}_{RI} \cup \mathcal{A}_{LR} \cup \mathcal{A}_{LIR}} W^i_{jk} + \sum_{j \in \mathcal{A}_L} W^l_{jk} + \sum_{j \in \mathcal{A}_I} W^i_{jk} + \sum_{j \in \mathcal{A}_R} W^r_{jk} - \tilde{E}_k$$

is a lower bound of the total work over time-interval $[t_1, t_2]$.

Hence, we have the following property:

Property 4. *If there exists a resource $k \in \mathcal{R}$ such that $\tilde{W}_k(t_1, t_2) > m_k(t_2 - t_1)$ then the instance is infeasible.*

Remark 1. *To derive better estimate of the slacks, we can substitute in (3.1) the quantity* $\{W_k(t_1,t_2) - W_{jk}(t_1,t_2)\}$ *by* $T_{jk}(t_1,t_2)$ *which represents the work calculated by* $(P1)$ *or* $(P2)$ *if we remove activity* j *from* $\mathcal{A}$*. Nevertheless, we empirically found that these time-bound adjustments have not improved the quality of the proposed lower bounds.*

3.4.6 Global revisited energetic reasoning

3.4.6.1 Mathematical model

Since Formulation $(P1)$ seeks for minimizing the work of activities scheduled in the left, right, or at the inside of an interval $[t_1, t_2]$ for a specified resource k, then we might have cases where an activity is scheduled in the left position for a resource k while it is scheduled in the right position for a resource $k' \neq k$. Hence, a better way to make feasibility tests consists in taking into account all the resources *simultaneously*. This approach is based on a formulation that is described as follows.

Let $E_k = m_k(t_2 - t_1) - (\sum_{j\in\mathcal{A}_L} W^l_{jk} + \sum_{j\in\mathcal{A}_R} W^r_{jk} + \sum_{j\in\mathcal{A}_I} W^i_{jk})$ be the work available on resource k for activities of the set $\mathcal{A}_{LI} \cup \mathcal{A}_{RI} \cup \mathcal{A}_{LR} \cup \mathcal{A}_{LIR}$.

The corresponding IP can be stated in the following way.

$$(\text{P3}) \quad \begin{aligned}
&\xi = \min \sum_{k\in\mathcal{R}} \epsilon_k && (3.23)\\
&\text{s.t.}\\
&x_j + z_j = 1, \quad \forall j \in \mathcal{A}_{LI} && (3.24)\\
&y_j + z_j = 1, \quad \forall j \in \mathcal{A}_{RI} && (3.25)\\
&x_j + y_j = 1, \quad \forall j \in \mathcal{A}_{LR} && (3.26)\\
&x_j + y_j + z_j = 1, \quad \forall j \in \mathcal{A}_{LIR} && (3.27)\\
&\sum_{j\in\mathcal{A}_\mathcal{L}} b_{jk}x_j \leq m^l_k, \quad \forall k \in \mathcal{R} && (3.28)\\
&\sum_{j\in\mathcal{A}_\mathcal{R}} b_{jk}y_j \leq m^r_k, \quad \forall k \in \mathcal{R} && (3.29)\\
&\sum_{j\in J^f_\mathcal{L}} b_{jk}\, x_j + \sum_{j\in J^f_\mathcal{R}} b_{jk}\, y_j + \sum_{j\in J^f_\mathcal{I}} b_{jk}\, z_j \leq m^f_k, \quad \forall k \in \mathcal{R} && (3.30)\\
&\sum_{j\in\mathcal{A}_\mathcal{L}} W^l_{jk}x_j + \sum_{j\in\mathcal{A}_\mathcal{R}} W^r_{jk}y_j + \sum_{j\in\mathcal{A}_\mathcal{I}} W^i_{jk}z_j - \epsilon_k \leq E_k, \quad \forall k \in \mathcal{R} && (3.31)\\
&x_j, y_j, z_j \in \{0,1\}, \quad \forall j \in \mathcal{A}_{LI} \cup \mathcal{A}_{RI} \cup \mathcal{A}_{LR} \cup \mathcal{A}_{LIR} && (3.32)\\
&\epsilon_k \geq 0, \quad \forall k \in \mathcal{R} && (3.33)
\end{aligned}$$

Constraints (3.24)-(3.30) have the same signification as in $(P1)$. Constraint (3.31) is added to check whether it is possible to schedule the activities such that the sum of their works do not exceed the total work available on each resource k. The slack variables ϵ_k are used to enforce this latter restriction. Indeed, ϵ_k is equal to zero if and only if the total work available on resource k is enough to schedule all the activities.

With this new mathematical model, we make only one feasibility test (and not K tests as in the revisited energetic reasoning). This new feasibility condition may be stated as follows.

Property 5. *If $\xi > 0$ then the instance is infeasible.*

Proof. Obvious. $\square$

In the sequel, we refer to this approach as the *Global revisited energetic reasoning*.

Example 4. *Consider a project with 5 activities, represented by Figure 3.5, and two resources with capacities equal to $B_1 = 7$ and $B_2 = 11$.*

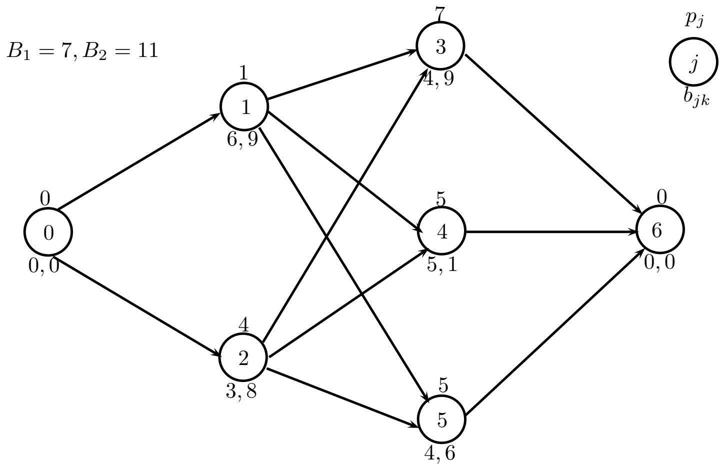


Figure 3.5: Global revisited energetic reasoning project example

The classical energetic reasoning-based lower bound is equal to 16. For this value, the revisited energetic reasoning-based lower bound does not detect an infeasibility.

The computation of the t_1 and t_2 values yields:

- $t_1 \in \{0, 4, 5, 8, 9, 11, 16\}$
- $t_2 \in \{0, 1, 4, 5, 9, 11, 16\}$

For the time-interval $[4, 16]$, we have $\mathcal{A}_{LR} = \{3\}$ and $\mathcal{A}_{LIR} = \{4, 5\}$. Solving the formulation (P3) over this time-interval gives $x_6 = y_4 = z_5 = 1$ and $\epsilon_1 = 1$, $\epsilon_2 = 0$. As a result, we detect an infeasibility. Finally, we find that the global revisited energetic reasoning-based lower bound is equal to 17.

3.4.6.2 Adjustments based on reduced costs

We can make adjustments by solving the linear relaxation of the previous formulation. For this, we consider the reduced costs of the non-bases variables. We distinguish two cases:

1. The variable is equal to 1 and its reduced cost is strictly negative: if the variable is forced to be 0, then the objective function increases and consequently the solution becomes infeasible. Then, we have:
   (a) If $x_j = 1$ then activity j must be at the left position, so $d_j \leftarrow \min(d_j, t_1 + p_j)$.
   (b) If $y_j = 1$ then activity j must be at the right position, so $r_j \leftarrow \max(r_j, t_2 - p_j)$.
   (c) If $z_j = 1$ then activity j must be in the inside, so $r_j \leftarrow \max(r_j, t_1 + 1)$ and $d_j \leftarrow \min(d_j, t_2 - 1)$.
2. The variable is equal to 0 and its reduced cost is strictly positive: if the variable is forced to be 1, then the objective function increases and consequently the solution becomes infeasible. Then, we have:
   (a) If $x_j = 0$ then activity j can not be at the left position, so $r_j \leftarrow \max(r_j, t_1 + 1)$.
   (b) If $y_j = 0$ then activity j can not be at the right position, so $d_j \leftarrow \min(d_j, t_2 - 1)$.
   (c) If $z_j = 0$ then activity j can not be in the inside, so:
      - If $j \in \mathcal{A}_{LI}$ then $d_j \leftarrow \min(d_j, t_1 + p_j - 1)$.
      - If $j \in \mathcal{A}_{RI}$ then $r_j \leftarrow \max(r_j, t_2 - p_j + 1)$.

Remark 2. *We found empirically that these time-bound adjustments have not improved the quality of the proposed lower bounds.*

3.5 Energetic reasoning and Dual Feasible Functions

Definition 1. *A function f is said to be discrete dual feasible if for any discrete finite set S of nonnegative integers, we have:*

$$\sum_{x \in S} x \leq B \Rightarrow \sum_{x \in S} f(x) \leq f(B)$$

where B is a nonnegative integer.

Dual feasible functions (DFFs) have been extensively used for deriving lower bounds both for one- and two-dimensional bin packing problems (see Fekete and Schepers [47], Haouari and Gharbi [50], and Carlier et al. [26]). The main idea is based on using a DFF to transform the item height and width and then to compute a lower bound on the transformed instance. Fekete and Schepers [47] show that this latter bound is indeed valid for the genuine instance. Interestingly, it is possible to use a very similar idea within the framework of energetic reasoning for the RCPSP. To that aim, after deriving a reduced instance, we use an appropriate discrete DFF to generate a modified instance by transforming the resource requirements and the resource availabilities as well. Next, the capacity bound, the critical sequence bound, and the m-machine bound are computed in turn. Eventually, the largest bound is retained. We denote the resulting bound by DFF. Moreover, we implemented a second, and much simpler, DFF-based lower bound in the following way: we simply use the simple feasibility conditions (see Section 3.4.3) to detect an infeasibility of the transformed instance. We refer the derived lower bound by DFF^*.

Remark 3. *It is easy to see that the lower bound that is based on Property 1 reduces to a trivial case of DFF.*

Remark 4. *Following the idea of Tercinet and Néron [103], we can use the modified instance for time-bound adjustments.*

After performing extensive experiments to assess the computational performance of the DFFs proposed in Fekete and Schepers [47] and Carlier et al. [26], we found that the best performance is obtained through the combination of the two following DFFs.

The first one, f_1^s $(1 \le s \le B/2)$, was proposed in Carlier et al. [26]. It is defined as follows:

$$\begin{aligned} f_1^s : [0, B] &\to [0, M_B(B, I)] \\ x &\mapsto \begin{cases} M_B(B, I) - M_B(B - x, I) & \text{if } x > B/2 \\ 1 & \text{if } s \le x \le B/2 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

where $M_B(B, I)$ is the solution of the knapsack problem defined by items of the set $I = \{i \in [1, n] : s \le c_i \le B/2\}$ $(s = 1, \ldots, B/2)$, capacity B, and where the objective is to maximize the number of selected items.

Remark 5. *In a strict sense, f_1^s is a Data Dependent Feasible Function (see Carlier et al [26]).*

The second DFF that we used has been proposed in Fekete and Schepers [47] and is defined (in a slightly different way) as follows:

$$\begin{aligned} f_2^s : [0, B] &\to [0, B] \\ x &\mapsto \begin{cases} x & \text{if } \frac{x(s+1)}{B} \in \mathbb{Z} \\ \left\lfloor \frac{x(s+1)}{B} \right\rfloor \frac{B}{s} & \text{otherwise} \end{cases} \end{aligned}$$

with $s = 1, 2$.

3.6 Experimental results

To assess the performance of the proposed lower bounds, we considered *PSPLIP* the well-known set of benchmark instances proposed by Kolisch et al. [66] (see Section 2.6).

All the experiments were conducted on a personal computer Pentium IV 3 Ghz with 1 GB of RAM and running under Windows XP. All the bounds were coded in C language and all the LPs and IPs were solved using CPLEX 11. Moreover, for computing the energetic reasoning-based lower bounds, we considered the set Σ of intervals that is defined as follows: $\Sigma = \{(t_1, t_2); t_1 \in \Omega_1, t_2 \in \Omega_2\}$ where:

- $\Omega_1 = \{r_j, j = 1, \ldots, n\} \cup \{d_j - p_j, j = 1, \ldots, n\}$
- $\Omega_2 = \{d_j, j = 1, \ldots, n\} \cup \{r_j + p_j, j = 1, \ldots, n\}$

It is worth mentioning that, for the sake of efficiency, Σ includes fewer time-intervals than those previously considered in Baptiste et al. [12].

A preprocessing phase can be made in order to adjust (reduce) the resource capacities. This step is based on the following mathematical formulation:

$$\tilde{B}_k = \max \sum_{j=1}^{n} b_{jk} X_j \tag{3.34}$$

s.t.

$$\sum_{i=1}^{n} b_{ik} X_i \leq B_k \tag{3.35}$$

$$X_i + X_j \leq 2 - M_{ij}, \quad \forall i \in \mathcal{A}, j \in \mathcal{A} \tag{3.36}$$

$$X_j \in \{0, 1\}, \quad \forall j \in \mathcal{A} \tag{3.37}$$

where $M_{ij} = \begin{cases} 0 & \text{if there is intersection between the intervals}[r_i, d_i] \text{ and } [r_j, d_j] \\ 1 & \text{otherwise} \end{cases}$

Constraint (3.36) imposes that assigned activities i (which the value of their variables X_i is equal to 1) can all be placed in parallel.

The new resource capacities are $\tilde{B}_k$.

Remark 6. *The tests showed that the adjustment of the capacities is useful to improve the results but when the DFFs are used then this preprocessing phase is useless.*

3.6.1 First set of experiments: computational performance of the proposed lower bounds

We conducted a first set of experiments to assess the performance of the following seven lower bounds:

- CER: the classical energetic reasoning-based bound (see Section 3.2)
- EER: the exact revisited energetic reasoning-based bound (see Section 3.4.4)
- RER: the relaxed revisited energetic reasoning-based bound (see Section 3.4.5)
- GER: the global energetic reasoning-based bound (see Section 3.4.6)
- GER_LP: LP relaxed variant of GER.
- IER: the improved energetic reasoning bound that is based on Corollary 1 and where the feasibility test is based on the simple bound that is equal to $\max(LB_C, LB_{CS}, LB_P)$

- IER^*: the improved energetic reasoning bound that is based on Proposition 1.

All these bounds include time-bound adjustments described in Property 2, i.e., the feasibility tests are reiterated until neither further adjustments are achieved nor infeasibility is detected.

Tables 3.1-3.2 display a summary of the computational results that were obtained on the 30- and 60-activity sets of instances. For each lower bound, we provide: $MGAP$: the mean percentage deviation with respect to the best known upper bound, $MTime$: the mean CPU time (in seconds), $\#(LB = UB)$: number of times where the lower bound is equal to the best known upper bound, $\#(LB = \max LB)$: number of times where the lower bound is maximal, $\#(LB > CER)$: number of times where the lower bound outperforms the classical energetic reasoning-based bound, and TAD : total absolute deviation from CER.

	$MGAP$(%)	$MTime$(s)	$\#(LB = UB)$	$\#(LB = maxLB)$	$\#(LB > CER)$	TAD
CER	7.65	0.01	119	202	0	0
EER	4.99	0.55	275	370	181	692
RER	4.99	0.02	275	369	179	690
GER	**4.96**	**0.13**	277	375	190	701
GER_LP	4.99	0.14	275	369	179	690
IER	**4.42**	**0.01**	279	418	226	920
IER^*	**4.09**	**8.99**	295	479	277	1014

Table 3.1: Results of the bounds on KSD30

	$MGAP$(%)	$MTime$(s)	$\#(LB = UB)$	$\#(LB = maxLB)$	$\#(LB > CER)$	TAD
CER	7.96	0.03	56	160	0	0
EER	3.08	1.50	326	448	290	1713
RER	3.09	0.04	326	446	288	1711
GER	**3.07**	**0.31**	327	453	295	1719
GER_LP	3.09	0.35	326	446	288	1711
IER	**2.92**	**0.05**	330	461	303	1779
IER^*	**2.87**	**32.33**	332	476	317	1801

Table 3.2: Results of the bounds on KSD60

A first striking observation is that all the proposed enhanced energetic reasoning-based lower bounds consistently outperform the classical energetic reasoning-based lower bound. More precisely, we observe that the two variants of the reduced instance-based lower bounds IER^* and IER exhibit a very good performance. Indeed, IER^* yields the tightest average deviation and strictly dominates CER, while IER yields the second best average deviation but is extremely fast. Moreover, we observe that bounds EER, RER, GER, and GER_LP have a quite similar overall performance (even though, GER stands out marginally by its quality and RER is the quickest). Surprisingly, we observe that the 0-1 formulation-based global energetic reasoning bound GER is slightly faster than the LP-based variant GER_LP. This paradoxical behavior is due to the fact that the feasibility test implemented within GER is stronger and therefore fewer iterations are often required to detect infeasibility.

3.6.2 Second set of experiments: assessment of the impact of DFFs

We analyzed the performance of the two DFFs-based lower bounds DFF and DFF^*. The results are displayed in Tables 3.3-3.4.

	$MGAP$(%)	$MTime$(s)	$\#(LB = UB)$	$\#(LB = maxLB)$	$\#(LB > CER)$	TAD
CER	5.17	0.03	182	338	100	485
DFF	**2.89**	0.20	324	480	242	1068
DFF^*	**2.89**	**0.02**	324	480	242	1068

Table 3.3: Impact of DFFs on KSD30

	$MGAP$(%)	$MTime$(s)	$\#(LB = UB)$	$\#(LB = maxLB)$	$\#(LB > CER)$	TAD
CER	7.01	0.13	90	215	0	0
DFF	**2.42**	0.63	353	480	311	1760
DFF^*	**2.42**	**0.07**	353	480	311	1760

Table 3.4: Impact of DFFs on KSD60

We see from Tables 3.3-3.4, that the DFFs enable to derive tight bounds that outperform all the energetic based bounds. Furthermore, we observe that these bounds are very fast. In particular, DFF^* yields excellent bounds while being impressively fast.

Remark 7. *We also implemented a variant of GER with DFFs. However, we found that the performance of this bound is similar to the performance of DFF and DFF^*.*

3.6.3 Third set of experiments: performance of a shaving procedure

Shaving also called global operations is a general approach that has been introduced by Carlier and Pinson [31] and Martin and Shmoys [77] in the context of the job shop problem. This approach is based on deriving sufficient conditions for proving that no feasible schedule can exist which involve specific start times. Therefore, it aims at reducing the time-window widths of the activities. Caseau and Laburthe [33] implemented a shaving procedure for the RCPSP that requires enumerating all the possible start times for each activity, but they found that this strategy is very time consuming. We implemented a different version of the shaving procedure that is based on a dichotomous search procedure: at each step, we split the time-interval of each activity into two time-intervals. Then, we sequentially invoke the energetic reasoning feasibility tests for each interval, in turn. A pseudo-code of this procedure is described in Algorithm 2.

We denote the resulting bound SHV.

Tables 3.5-3.8 present the results of a comparative study of the performance of DFF^*, SHV as well as the column generation-based lower bound CG of Brucker and Knust [23] that is considered as one of the strongest lower bound for the RCPSP. It is noteworthy that the reported CPU times of CG were obtained on a Sun Ultra 2 workstation 167 MHz.

We see that:

Algorithm 2 Shaving procedure

```
1: repeat
2:   Energetic reasoning
3:   for all $j \in \mathcal{A}$ do
4:     compute $M_j = \left\lceil \frac{r_j + d_j - p_j}{2} \right\rceil$,
5:     Make feasibility tests on intervals $I_j^1 = [r_j, M_j + p_j - 1]$ and $I_j^2 = [M_j, d_j]$,
6:     if infeasibility is detected on $I_j^1$ and $I_j^2$ then
7:       increment the value of the trivial value and go to line 2
8:     else if infeasibility is detected only on $I_j^1$ then
9:       adjust the release date; $r_j \leftarrow M_j$
10:    else if infeasibility is detected only on $I_j^2$ then
11:      adjust the due date; $d_j \leftarrow M_j + p_j - 1$
12:    end if
13:  end for
14: until there is no infeasibility detected
```

	$MGAP(\%)$	$MTime(s)$	$\#(LB = UB)$	$\#(LB = maxLB)$
DFF^*	2.89	0.02	324	343
SHV	1.73	0.50	**366**	480
CG	1.5	0.4	318	-

Table 3.5: Results of the shaving on KSD30

- for all the sets of instances, SHV yields bounds that are much often equal to an upper bound than CG does
- for all the sets of instances, SHV yields bounds that are much often maximal than CG does
- for the 90- and 120-activity sets of instances, SHV exhibits a smaller average percentage gap than CG does.

Furthermore, we observe that DFF^* often dominates CG while being significantly faster.

We also tested the lower bounds on **BL** instances (proposed by Baptiste and Le Pape

	$MGAP(\%)$	$MTime(s)$	$\#(LB = UB)$	$\#(LB = maxLB)$
DFF^*	2.42	0.07	353	372
SHV	1.98	2.16	**362**	424
CG	1.85	5.00	342	423

Table 3.6: Results of the shaving on KSD60

	$MGAP(\%)$	$MTime(\text{s})$	$\#(LB = UB)$	$\#(LB = maxLB)$
DFF^*	1.75	0.32	358	400
SHV	**1.60**	4.78	**370**	439
CG	1.62	72.00	353	436

Table 3.7: Results of the shaving on KSD90

	$MGAP(\%)$	$MTime(\text{s})$	$\#(LB = UB)$	$\#(LB = maxLB)$
DFF^*	3.71	0.93	216	428
SHV	**3.46**	23.37	**235**	524
CG	3.62	355.00 (*)	214	478

(*)This average is calculated for the 481 instances of KSD120 which are calculated within a time limit of 1 hour

Table 3.8: Results of the shaving on KSD120

[9]) and **PACK** instances (proposed by Néron [87]). The results are reported in Tables 3.9-3.10.

For BL instances, the CER lower bound gives good results because the instances are highly cumulative. Thus, all other lower bounds have not improved the results. Only the shaving procedure ameliorated 6 instances among the 18 instances that have not been proved optimal with the CER lower bound.

	$MGAP(\%)$	$MTime(\text{s})$	$\#(LB = UB)$	$\#(LB = maxLB)$
CER	2.41	0.02	21	33
SHV	1.76	0.04	25	39

Table 3.9: Results of the bounds on BL instances

For PACK instances, the CERlower bound have a mean deviation of 9.35%. This deviation is reduced to 4.89% by the DFF lower bound. All other lower bounds (including the shaving lower bound SHV) show the same performance.

3.7 Conclusion

In this chapter, we presented three classes of lower bounds that are based on the concept of energetic reasoning. The first class includes bounds that are based on the concept of reduced instances. We provided evidence that this concept proves useful to generate high quality lower bounds. The second class includes bounds that are based on the so-called revisited energetic reasoning that was initially introduced for parallel machine scheduling problems. We proposed several non trivial generalizations of this concept. In particular, we introduced a new global feasibility test that considers all the resources simultaneously. Finally, the last class of bounds are based on discrete dual

	$MGAP(\%)$	$MTime(s)$	$\#(LB = UB)$	$\#(LB = maxLB)$
CER	9.35	0.01	13	31
DFF	4.89	0.00	19	55

Table 3.10: Results of the bounds on PACK instances

feasible functions. Our computational results provide evidence that a deceptively simple DFF-based lower bound is competitive with a state-of-the-art lower bound while being extremely fast. Furthermore, we found that an effective shaving procedure enables to derive a highly competitive lower bound that often outperforms the best bound from the literature while being significantly simpler.

Chapter 4

A Preemptive Bound for the Resource Constrained Project Scheduling Problem

4.1 Introduction

Numerous lower bounds have been proposed for the RCPSP. Depending on how the checking procedure (for proving infeasibility) is carried out, numerous destructive lower bounds have been proposed so far for the RCPSP. In particular, we quote the paper by Klein and Scholl [63], where the idea of destructive bound was originally introduced, and those by Brucker and Knust [23] and Baptiste and Demassey [10] where destructive lower bounds, that are the best lower bounds which are currently available for the RCPSP, are described. These latter bounds are derived through the resolution of several large-scale linear programs by column generation. Actually, these bounds require intensive constraint propagation techniques-based preprocessing as well as an involved implementation of column generation. We note also the recent work of Schutt et al. [96] who presented several procedures based on constraint programming. Experimental results demonstrate the very good performance of the proposed approach which delivered proven optimal solutions for several open instances.

In this chapter, we propose a destructive lower bound where the checking procedure is based on solving an enhanced LP relaxation of the RCPSP. Even though the basic features of our approach are classic, they are enriched with some novel ideas which make it attractive. Indeed, starting from a relaxation from the literature, that is shown to perform poorly, we enhanced it with a variety of valid inequalities that enforce precedence relationships, resource constraints, and nonpreemption. With few exceptions, all these valid inequalities are original or dominate previously proposed cuts. Furthermore, a distinctive feature of the proposed lower bound is that it requires the solution of a *compact* LP (that is, having a polynomial number of variables and constraints) and therefore does not require the implementation of a sophisticated column generation algorithm. A nice consequence of this compactness feature is that our approach is scalable and therefore can be used for large scaled instances.

The remainder of the chapter is organized as follows. In Section 4.2, we introduce a basic LP relaxation that was previously proposed in the literature and that will be used as a starting model for building an enhanced LP relaxation through appending several new valid constraints. To that aim, we begin with by presenting in Section 4.3 a new precedence constraint. Incompatible activity sets-based cuts are presented in Sections 4.4. Nonpreemptive and Energetic reasoning-based cuts are presented in Sections 4.5 and 4.6, respectively. Finally, the results of our extensive computational experiments are presented in Section 4.7.

It is worth noting that most of the material presented in this chapter has been the subject of two conferences [71, 70] and has been submitted as a journal paper [52].

4.2 A basic preemptive formulation

We introduce a basic relaxation of the RCPSP that was originally proposed by Carlier and Néron [29]. This relaxation aims at checking whether there exists a feasible schedule having a makespan shorter than or equal to the trial value C. It is worth emphasizing that instead of considering an exact RCPSP feasibility problem (this latter problem being $\mathcal{NP}$-hard) we focus on a relaxation of the RCPSP, where resource and precedence constraints are partially considered, that is formulated as a linear program. Toward this end, we introduce the following notation. Let $\{t_1, \ldots, t_{L+1}\}$ be the set containing all release dates and deadlines of all activities ranked in increasing order. A time interval $I_l = [t_l, t_{l+1})$ is defined for each $l \in \mathcal{L}$ where $\mathcal{L} = \{1, \ldots, L\}$. Hence, the time-horizon $[0, C]$ (recall that C is a trial value of the makespan) is split into L consecutive time-intervals whose time-bounds correspond either to a release date or a deadline. Moreover, we define the following sets:

- $\mathcal{I}^j = \{l \in \mathcal{L} : I_l \subseteq [r_j, d_j]\}$ the set of indices of time-intervals during which activity j can be processed,
- $\mathcal{I}_i^j = \{l \in \mathcal{L} : \max(r_i, r_j) \leq t_l < \min(d_i, d_j)\}$ the set of indices of time-intervals during which both activities i and j can be in progress simultaneously,
- $\mathcal{A}^l = \{j \in \mathcal{A} : I_l \subseteq [r_j, d_j]\}$ the set of activities that can be in progress during time-interval I_l.

Also, we set $\Delta_l = t_{l+1} - t_l$ for $l \in \mathcal{L}$, $v = \max_{j \in \mathcal{A}}(d_j - p_j - r_j + 1)$, and $\rho = \max_{j \in \mathcal{A}} \left|\mathcal{I}^j\right|$. These latter two parameters represent the maximum number of feasible starting times of an activity, and the maximum number of time-intervals during which an activity might be processed, respectively.. Furthermore, we define the continuous decision variable x_{jl} as the amount of time that is allocated to process activity $j \in \mathcal{A}$ during the time-interval $I_l \in \mathcal{I}^j$. Thus, the feasibility problem proposed by Carlier and Néron [29] reads as follows:

$$
(P4) \quad
\begin{aligned}
&\text{Find } x && (4.1)\\
&\text{s.t.} && \\
&\sum_{l \in I^j} x_{jl} = p_j, \quad \forall j \in \mathcal{A} && (4.2)\\
&x_{jl} \leq \Delta_l, \quad \forall j \in \mathcal{A}, \forall l \in \mathcal{I}^j && (4.3)\\
&\sum_{j \in \mathcal{A}^l} b_{jk} x_{jl} \leq B_k \Delta_l, \quad \forall k \in \mathcal{R}, \forall l \in \mathcal{L} && (4.4)\\
&\sum_{s \in \mathcal{I}^i / s \leq l} \frac{x_{is}}{p_i} \geq \sum_{s=\tau / t_\tau = r_j}^{l} \frac{x_{js}}{p_j}, \quad \forall (i,j) \in A, \forall l \in \mathcal{I}_i^j && (4.5)\\
&x_{jl} \geq 0, \quad \forall j \in \mathcal{A}, \forall l \in \mathcal{I}^j && (4.6)
\end{aligned}
$$

Constraints (4.2) state that each activity j must be processed within its specified time-window $[r_j, d_j]$. Constraints (4.3) require that any activity j could not be processed more than once at any time. Constraints (4.4) enforce that for any resource k and at any time-interval I_l the total resource requirements do not exceed the resource capacity. Constraints (4.5) can be viewed as "pseudo-precedence" constraints. They enforce that if i is a predecessor of j then the total fraction of i that is processed from its release date r_i up to the time-bound t_{l+1} is larger than or equal to the fraction of the successor activity j that is processed from its release date r_j up to the same time-bound t_{l+1}.

It is worth mentioning that, in a strict sense, $(P4)$ does not correspond to the preemptive relaxation of the RCPSP. This is mainly due to the fact that (4.4) can be viewed as a relaxed version of the cumulative resource constraints.

Property 6. *If for a given trial value C of the makespan, $(P4)$ is infeasible then $C+1$ is a valid lower bound on the optimal makespan.*

In the sequel, we denote LB_{CN} the destructive bound that is based on Property 6. Interestingly, even though LB_{CN} has been proposed few years ago, its computational performance has never been thoroughly assessed. Thus, we conducted a computational analysis of LB_{CN} and compared it with the state-of-the-art lower bound of Brucker and Knust [23] (in the sequel, we shall refer to this latter bound by LB_{BK}). In our implementation, we considered as an initial trial value the so-called critical path bound that is defined as follows:

$$LB_{CP} = l(0, n+1)$$

We considered a subset of 326 nontrivial 60- and 90- activity instances from the well-known PSPLIB testbed proposed by Kolisch et al. [66] for which LB_{CP} is strictly smaller than the best known upper bound. A summary of the results is displayed in Table 4.1. In this table, each entry represents the percentage of times the equality (or inequality) that is stated in the first column of the corresponding row holds for the different problem sets, respectively. Not surprisingly, we observe that LB_{CN} exhibits a poor performance and is largely dominated by LB_{BK}. Actually, we found that LB_{CN} is strictly outperformed by LB_{BK} for 81.6% of the instances.

	60-activity	90-activity
$LB_{CN} = LB_{BK}$	29	31
$LB_{CN} < LB_{BK}$	154	112
$LB_{CN} > LB_{BK}$	0	0

Table 4.1: Comparison between LB_{CN} and LB_{BK} on nontrivial instances

In the sequel, we shall gradually append several valid constraints to ($P4$) and provide evidence that these additional cuts will have a dramatic effect on the performance of the resulting destructive bound.

4.3 A precedence-based cut

Let x be a *nonpreemptive* schedule and p_{jl} be the mandatory part of an activity $j \in \mathcal{A}$ over an interval I_l. We define $\alpha_{jl} = (t_l + p_{jl})x_{jl} - \frac{1}{2}p_{jl}(p_{jl}+1)$ and $\beta_{jl} = (t_{l+1} - 1 - p_{jl})x_{jl} + \frac{1}{2}p_{jl}(p_{jl}+1)$, for each activity $j \in \mathcal{A}$ and each time-interval I_l, $l \in \mathcal{I}^j$. Also, we denote by $s_j \in [r_j, d_j - p_j]$ the starting time of activity j. Thus, j is processed during the consecutive time points $s_j, \dots, (s_j + p_j - 1)$. Then, the mean time point during which the resources are allocated to j is $\tilde{s}_j = s_j + \dfrac{p_j - 1}{2}$ for $j \in \mathcal{A}$. We have:

Proposition 2. $\dfrac{\sum\limits_{l \in \mathcal{I}^j} \alpha_{jl}}{p_j} \le \tilde{s}_j \le \dfrac{\sum\limits_{l \in \mathcal{I}^j} \beta_{jl}}{p_j}$ *for $j \in \mathcal{A}$*

Proof. First, note that in this case the x_{jl}'s are integers. Assume that $x_{jl} > 0$ and that $\{s_j, \dots, s_j + p_j - 1\} \cap I_l = \{s_j^l, \dots, s_j^l + x_{jl}\}$. Thus, we have the following inequalities:

$$(S1)\begin{cases} t_l \le s_j^l \\ t_l + 1 \le s_j^l + 1 \\ \dots \\ t_l + p_{jl} - 1 \le s_j^l + p_{jl} - 1 \\ t_l + p_{jl} \le s_j^l + p_{jl} \\ \dots \\ t_l + p_{jl} \le s_j^l + x_{jl} - 1 \end{cases}$$

$$(S2)\begin{cases} s_j^l \leq t_{l+1} - p_{jl} - 1 \\ \dots \\ s_j^l + x_{jl} - p_{jl} - 1 \leq t_{l+1} - p_{jl} - 1 \\ s_j^l + x_{jl} - p_{jl} \leq t_{l+1} - p_{jl} \\ s_j^l + x_{jl} - p_{jl} + 1 \leq t_{l+1} - p_{jl} + 1 \\ \dots \\ s_j^l + x_{jl} - 1 \leq t_{l+1} - 1 \end{cases}$$

Summing all inequalities in (S1) and (S2), we get:

$$\alpha_{jl} \leq \sum_{q=0}^{x_{jl}-1} (s_j^l + q) \leq \beta_{jl}, \quad \forall j \in \mathcal{A}, \forall l \in \mathcal{I}^j \tag{4.7}$$

Summing up all inequalities (4.7) for $l \in \mathcal{I}^j$, we get:

$$\sum_{l \in \mathcal{I}^j} \alpha_{jl} \leq s_j + \ldots + (s_j + p_j - 1) \leq \sum_{l \in \mathcal{I}^j} \beta_{jl}, \quad \forall j \in \mathcal{A} \tag{4.8}$$

Finally, this yields

$$\frac{\sum_{l \in \mathcal{I}^j} \alpha_{jl}}{p_j} \leq \tilde{s}_j \leq \frac{\sum_{l \in \mathcal{I}^j} \beta_{jl}}{p_j}, \quad \forall j \in \mathcal{A}$$

□

Furthermore, for each $j \in \mathcal{A}$ and $s_j \in [r_j, d_j - p_j]$, we compute the corresponding values α_j, β_j, and $\tilde{s}_j$, and we define the slack variables $\lambda_j(s_j) = \tilde{s}_j - \frac{\sum_{l \in \mathcal{I}^j} \alpha_{jl}}{p_j}$ and $\mu_j(s_j) = \frac{\sum_{l \in \mathcal{I}^j} \beta_{jl}}{p_j} - \tilde{s}_j$, respectively.

Lemma 1. *The following inequality*

$$\frac{\sum_{l \in \mathcal{I}^j} \beta_{jl}}{p_j} - \frac{\sum_{l \in \mathcal{I}^i} \alpha_{il}}{p_i} \geq \min_{s_i \in [r_i, d_i - p_i], s_j \in [r_j, d_j - p_j] : s_i + p_i \leq s_j} (\lambda_i(s_i) + \mu_j(s_j)) + \frac{p_i}{2} + \frac{p_j}{2}, \quad \forall (i,j) \in A \tag{4.9}$$

is valid.

Proof. Assume that $(i, j) \in A$. If activity i starts at s_i and activity j starts at s_j then we have:

$$s_i + p_i \leq s_j \Rightarrow \tilde{s}_i + \frac{p_i}{2} \leq \tilde{s}_j - \frac{p_j}{2}$$
$$\Rightarrow \lambda_i(s_i) + \frac{\sum_{l\in\mathcal{I}^i} \alpha_{il}}{p_i} + \frac{p_i}{2} \leq \frac{\sum_{l\in\mathcal{I}^j} \beta_{jl}}{p_j} - \frac{p_j}{2} - \mu_j(s_j)$$

Hence, we derive the valid inequality

$$\frac{\sum_{l\in\mathcal{I}^j} \beta_{jl}}{p_j} - \frac{\sum_{l\in\mathcal{I}^i} \alpha_{il}}{p_i} \geq (\lambda_i(s_i) + \mu_j(s_j)) + \frac{p_i}{2} + \frac{p_j}{2} \tag{4.10}$$

By considering all possible (compatible) starting times, we derive the valid inequality (4.9).

□

Clearly, for each $j \in \mathcal{A}$ and $s_j \in [r_j, d_j - p_j]$, the computation of $\lambda_j(s_j)$ and $\mu_j(s_j)$ requires $O(n\rho)$-time. Also, for each $(i,j) \in A$, the computation of the RHS of the corresponding inequality (4.9) requires $O(v^2)$-time. Thus, the overall effort for generating (4.9) is $O(n\rho + |A|\, v^2)$.

4.4 Incompatible activity sets-based cuts

Now, we turn our attention to valid cuts that are based on incompatible activity sets. First, we describe three cuts from the literature. Next, we introduce the so-called clique cuts together with the associated separation procedure. Finally, we show how to use the concept of dual feasible function to derive from the capacity constraints (4.4) several valid inequalities.

4.4.1 Three cuts from the literature

4.4.1.1 Parallel machine cuts

First, we recall (see Section 2.3.1) the definition of the sets $\mathcal{E}_k^m$. For each resource $k \in \mathcal{R}$, and each integer m, these sets are defined as $\mathcal{E}_k^m = \{j \in \mathcal{A} : b_{jk} \geq \left\lfloor \frac{B_k}{m+1} \right\rfloor + 1\}$. Following Carlier and Latapie [27], we observe that in any feasible solution there must be at most m activities from $\mathcal{E}_k^m$ that can be processed in parallel. Hence, a valid constraint is:

$$\sum_{j\in\mathcal{A}^l\cap\mathcal{E}_k^m} x_{jl} \leq m\Delta_l, \quad \forall l \in \mathcal{L}, \forall k \in \mathcal{R} \tag{4.11}$$

In our tests, and for the sake of computational efficiency, we restrict these constraints to $m \in \{1, 2\}$.

It is easy to see that for each $k \in \mathcal{R}$, and each integer m, (4.11) can be derived in $O(n)$-time.

4.4.1.2 Minimal forbidden cuts

Furthermore, Nabeshima [81] introduced a more general concept of forbidden set of activities. This concept refers to activity subsets that include activities that cannot be in progress simultaneously. Only minimal forbidden sets are considered, i.e., all subsets of a minimal forbidden set are not a forbidden set.

Let ϕ be a minimal forbidden set, and $|\phi|$ its cardinality. Then, a valid constraint is:

$$\sum_{j \in \mathcal{A}^l \cap \phi} x_{jl} \leq (|\phi| - 1)\Delta_l, \quad \forall l \in \mathcal{L}, \forall \phi \in \Phi \tag{4.12}$$

where Φ is the set of all minimal forbidden sets.

Notice that the number of minimal forbidden sets may be large (see Artigues et al. [5]). Therefore, in the sequel, we shall restrict constraints (4.12) only to the minimal forbidden sets of cardinality 3. Hence, generating these cuts requires $O(n^3)$ time.

Remark 8. *Notice that all the sets $\mathcal{E}_k^2$ of cardinality 3 (see Constraint 4.11) are included in the minimal forbidden sets of cardinality 3. Thus, some cuts generated by (4.11) are also generated by (4.12). However, in general, a set $\mathcal{E}_k^2$ may be of cardinality larger than 3.*

4.4.1.3 Job shop cuts

In the following, we present an adaptation of the *Two-job* cuts introduced by Applegate and Cook [4] for the resolution of the job shop scheduling problem.

Let i and j be two activities in disjunction on a resource, i.e., there exists a resource $k \in \mathcal{R}$ such that $b_{ik} + b_{jk} > B_k$ then we know that i precedes j or j precedes i. We have the following constraint:

$$(r_i + p_i - r_j)s_i + (r_j + p_j - r_i)s_j \geq p_i p_j + r_i p_j + r_j p_i \quad \text{if } r_i + p_i \geq r_j \text{ and } r_j + p_j \geq r_i \tag{4.13}$$

Proof. If i precedes j, then:

$$s_i \geq r_i$$
and
$$s_j \geq r_i + p_i$$

If we multiply the first inequality by $r_i + p_i - r_j$ and the second one by $r_j + p_j - r_i$ and do the addition of the two resulting inequalities, we obtain the constraint (4.13). In the same way, if j precedes i then we prove the result by permuting the indices i and j. □

Let $\tilde{\lambda}_j = \min_{s_j \in [r_j, l_j]} (\tilde{s_j} - \frac{\alpha_j}{p_j})$ and $\tilde{\mu}_j = \min_{s_j \in [r_j, l_j]} (\frac{\beta_j}{p_j} - \tilde{s_j})$. We have the following inequality:

$$\frac{\alpha_j}{p_j} + \tilde{\lambda}_j \leq \tilde{s}_j \leq \frac{\beta_j}{p_j} - \tilde{\mu}_j$$
$$\Rightarrow \frac{\alpha_j}{p_j} + \tilde{\lambda}_j - \frac{p_j - 1}{2} \leq s_j \leq \frac{\beta_j}{p_j} - \tilde{\mu}_j - \frac{p_j - 1}{2} \tag{4.14}$$

Moreover, let $\sigma_{ij} = r_i + p_i - r_j$. Using the last inequality, the constraint (4.13) becomes:

$$\frac{\sigma_{ij}}{p_i}\beta_i + \frac{\sigma_{ji}}{p_j}\beta_j \geq p_i p_j + \frac{p_i^2 + p_j^2 + r_j p_i + r_i p_j - p_i - p_j}{2} + \sigma_{ij}\tilde{\mu}_i + \sigma_{ji}\tilde{\mu}_j \tag{4.15}$$

In the same way, we can prove the following constraint:

$$\rho_{ij} c_i + \rho_{ji} c_j \leq d_i p_j + d_j p_i - p_i p_j \quad \text{si } d_j \geq d_i - p_i \text{ and } d_i \geq d_j - p_j \tag{4.16}$$

where $\rho_{ij} = d_j - d_i + p_i$.

We know also that c_j is equal to $s_j + p_j$. Thus, the last relation becomes:

$$\rho_{ij} s_i + \rho_{ji} s_j \leq d_i p_i + d_j p_j - p_i p_j - p_i^2 - p_j^2 \tag{4.17}$$

Using the inequality (4.14) and the constraint (4.17), we obtain the following constraint:

$$\frac{\rho_{ij}}{p_i}\alpha_i + \frac{\rho_{ji}}{p_j}\alpha_j \leq -p_i p_j + \frac{d_i p_i + d_j p_j + d_i p_j + d_j p_i - p_i^2 - p_j^2 - p_i - p_j}{2} - \rho_{ij}\tilde{\lambda}_i - \rho_{ji}\tilde{\lambda}_j \tag{4.18}$$

4.4.2 Maximum number of activities in parallel

Next idea we have investigated consists in determining for each time-interval the maximum number of activities that can be processed in parallel. For each time-interval I_l, let us define $(P5)$ the following ILP formulation:

$$\mu(\mathcal{A}^l) = \max \sum_{j \in \mathcal{A}^l} y_j \tag{4.19}$$

$$(\text{P5}) \quad \text{s.t.}$$
$$\sum_{j \in \mathcal{A}^l} b_{jk} y_j \leq B_k, \quad \forall k \in \mathcal{R} \tag{4.20}$$
$$y_j \in \{0, 1\}, \quad \forall j \in \mathcal{A}^l \tag{4.21}$$

$\mu(\mathcal{A}^l)$ represents the maximum number of activities that may be in parallel execution over the interval $[t_l, t_{l+1}]$. Then a valid constraint is:

$$\sum_{j \in \mathcal{A}^l} x_{jl} \leq \mu(\mathcal{A}^l)(t_{l+1} - t_l), \quad \forall l \in \mathcal{L} \tag{4.22}$$

Remark 9. *As the resolution of the ILP formulation (P5) may be time consuming, constraints (4.22) will not be experimentally tested.*

4.4.3 Clique cuts

In this section, we introduce a valid inequality that is based on subsets of mutually incompatible activities. To that aim, we define $G^{'} = (V, E)$ an undirected incompatibility graph, presented for the RCPSP by Mingozzi et al. [78], where each node i represents an activity $i \in \mathcal{A}$ and where each edge $\{i, j\} \in E$ is associated with a pair $\{i, j\}$ of activities that cannot both be processed simultaneously. That is:

$$\{i, j\} \in E \Longleftrightarrow \begin{cases} (1)\ i \text{ is a predecessor of } j \text{ (or } j \text{ is a predecessor of } i) \\ \text{or} \\ (2)\ b_{ik} + b_{jk} > B_k \text{ for some } k \in \mathcal{R} \end{cases}$$

It is noteworthy that Condition (1) does not necessarily require an *immediate* precedence relationship.

Let $\mathcal{C}$ denote the set of cliques of $G^{'}$. Clearly, a clique $Cl \in \mathcal{C}$ includes a set of mutually exclusive activities. Hence, the following cut

$$\sum_{j \in \mathcal{A}^l \cap Cl} x_{jl} \leq \Delta_l, \quad \forall l \in \mathcal{L}, \forall Cl \in \mathcal{C} \tag{4.23}$$

is valid.

Clearly, it is sufficient to consider in (4.23) only the (nondominated) inclusionwise maximal cliques.

Since (4.23) includes a very large (exponential) number of constraints, then the feasibility LP model is solved using a constraint generation approach where violated clique constraints are iteratively generated on the fly. Hence, this approach requires iteratively solving $|\mathcal{L}|$ separation problems of the form:

(SP^l): *Given* $(\bar{x}_{jl})_{j=1,\ldots,n}$ *the solution of* $LP0$, *find* $Cl \in \mathcal{C}$ *such that* $\sum_{A_j \in \mathcal{A}^l \cap Cl} \bar{x}_{jl} > \Delta_l$ *or verify that no such clique exists.*

Thus, (SP^l) amounts to finding a clique $Cl \in \mathcal{C}$ such that $\sum_{A_j \in \mathcal{A}^l \cap Cl} \bar{x}_{jl}$ is maximal.

Therefore, we see that (SP^l) requires solving a *maximum weight clique problem* (MWCP) in the subgraph $G^l = (V^l, E^l)$ of $G^{'}$ (where each node j from V^l represents an activity $A_j \in \mathcal{A}^l$) and where each node is assigned a weight $\bar{x}_{jl}$. This latter problem is notoriously known to be intractable. In our implementation, we used the exact algorithm of Östergård (2001) to solve the MWCP.

Alternatively, we tested a much faster (and significantly simpler) heuristic strategy where a restricted number of clique cuts are only generated in a preprocessing step. Hence, in so doing, we avoid invoking a time consuming dynamic constraint generation procedure. It is noteworthy that since the clique cuts are now generated prior to computing the x_{jl}'s variables, then we cannot emphasize inclusionwise maximal cliques. Instead, the implemented heuristic is based on generating the cliques that are maximal w.r.t cardinality. More precisely, we implemented the following strategy:

- **Step 1:** Invoke an exact algorithm to generate the cliques that are maximal w.r.t cardinality
  (In our implementation, we used the algorithm of Östergård (2002) and we set the CPU time limit to 300 s).
- **Step 2:** Sort the generated cliques in decreasing cardinality
- **Step 3:** Select the largest #*CC* cliques and append them to the feasibility model (in our implementation, we empirically set the parameter #*CC* to 80)

4.4.4 Dual feasible functions-based cuts

Given a feasible schedule x, and a DFF f, we denote by $\mathcal{A}_t^l$ the subset of activities that are in progress at time $t \in I_l$ **(**$l \in \mathcal{L}$). We have:

$$\sum_{A_j \in \mathcal{A}_t^l} b_{jk} \leq B_k, \quad \forall t \in I_l, \ \forall l \in \mathcal{L}, \ \forall k \in \mathcal{R} \tag{4.24}$$

Since f is a DFF, then we get:

$$\sum_{A_j \in \mathcal{A}_t^l} f(b_{jk}) \leq f(B_k), \quad \forall t \in I_l, \ \forall l \in \mathcal{L}, \ \forall k \in \mathcal{R} \tag{4.25}$$

After summing up all inequalities for $t \in I_l$, we derive:

$$\sum_{A_j \in \mathcal{A}^l} f(b_{jk}) x_{jl} \leq f(B_k) \Delta_l, \quad \forall l \in \mathcal{L}, \forall k \in \mathcal{R} \tag{4.26}$$

Hence, given a set $\mathcal{F} = \{f_h\}_{h=1,\ldots,\xi}$ of DFFs, we can derive from (4.4) the following valid inequalities:

$$\sum_{A_j \in \mathcal{A}^l} f_h(b_{jk}) x_{jl} \leq f_h(B_k) \Delta_l, \quad \forall l \in \mathcal{L}, \forall k \in \mathcal{R}, \forall f_h \in \mathcal{F} \tag{4.27}$$

For the sake of computational efficiency, we performed extensive computational experiments to identify an effective set of DFFs. We found that a good performance is obtained by implementing the same DFFs f_1^s of Fekete and Schepers [47] and f_2^s of Carlier et al. [26] described in Section 3.5.

We note that each cut that is based on f_1^s can be generated in $O(n)$-time for each $k \in \mathcal{R}$. Also, a cut that is based on f_2^s can be generated in $O(n \log n)$-time for each $k \in \mathcal{R}$ (because f_2^s needs the resolution of a knapsack problem which is obtained by trivially ranking the items in nondecreasing weights).

4.5 Nonpreemptive cuts

In this section, we propose four families of cuts that are based on the requirement that a feasible RCPSP schedule is nonpreemptive.

4.5.1 First family

Lemma 2. *The following constraints*

$$\frac{\sum_{l=p+1}^{q-1} x_{jl}}{\sum_{l=p+1}^{q-1} \Delta_l} \geq \frac{x_{jp}}{\Delta_p} + \frac{x_{jq}}{\Delta_q} - 1, \quad \forall j \in \mathcal{A}, \forall p,\, q \in \mathcal{I}^j \text{ such that } p+1 < q \tag{4.28}$$

are valid.

Proof. Assume that x represents a nonpreemptive schedule. We consider two cases:

- Case 1: $\frac{x_{jp}}{\Delta_p} + \frac{x_{jq}}{\Delta_q} > 1$. Thus, activity j is both processed during the intervals I_p and I_q (with $p+1 < q$). Since x is nonpreemptive, then j is necessarily continuously processed during all the intermediate intervals $I_{p+1}, \ldots, I_{q-1}$ as well. Hence, $\frac{\sum_{l=p+1}^{q-1} x_{jl}}{\sum_{l=p+1}^{q-1} \Delta_l} = 1$ and therefore the constraint is valid.

- Case 2: $\frac{x_{jp}}{\Delta_p} + \frac{x_{jq}}{\Delta_q} \leq 1$. Then, the constraint is trivially valid. □

Each constraint (4.28) can be generated in $O(\rho)$-time.

4.5.2 Second family

Let $\mathcal{R}^j = \{p, p+1, \ldots, q\}$ be the set of indices of time intervals during which activity $A_j \in \mathcal{A}$ must be in progress if it starts at its release date r_j. That is,

$$l \in \mathcal{R}^j \Leftrightarrow [r_j, r_j + p_j] \cap I_l \neq \emptyset.$$

For convenience, we assume that $\left|\mathcal{R}^j\right| > 1$ (see Figure 4.1).

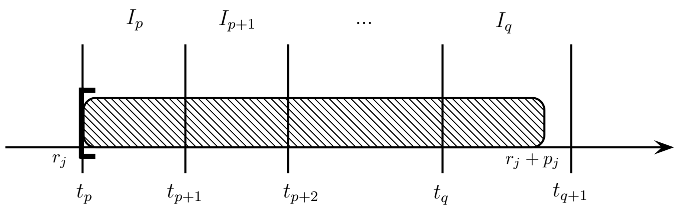


Figure 4.1: Illustration of nonpreemptive cuts (second family)

Lemma 3. *We have:*

$$\frac{x_{jp}}{\Delta_p} \leq \ldots \leq \frac{x_{j,q-1}}{\Delta_{q-1}} \leq \epsilon_j \frac{x_{jq}}{\Delta_q}, \quad \forall j \in \mathcal{A} \tag{4.29}$$

where $\epsilon_j = \dfrac{\Delta_q}{r_j + p_j - t_q}$

Proof. We consider two cases:

Case 1: There exists $h \in [p, q-1]$ such that $\dfrac{x_{jh}}{\Delta_h} > 0$. Hence, $\dfrac{x_{jl}}{\Delta_l} = 1$ for $l = h+1, \ldots, q-1$. Also, $x_{jq} \geq r_j + p_j - t_q$. Thus, the inequalities are valid.

Case 2: $\dfrac{x_{jl}}{\Delta_l} = 0$, for $l = p, \ldots, q-1$. In this case, the inequality reduces to $\epsilon_j \dfrac{x_{jq}}{\Delta_q} \geq 0$ and is therefore trivially valid. □

Furthermore, let $\mathcal{D}^j = \{p, p+1, \ldots, q\}$ be the set of indices of time intervals during which activity $j \in \mathcal{A}$ must be in progress if it terminates at its deadline d_j. That is,

$$l \in \mathcal{D}^j \Leftrightarrow [d_j - p_j, d_j] \cap I_l \neq \emptyset.$$

Lemma 4. *We have:*

$$\epsilon_j' \frac{x_{jp}}{\Delta_p} \geq \frac{x_{j,p+1}}{\Delta_{p+1}} \geq \ldots \geq \frac{x_{jq}}{\Delta_q}, \quad \forall j \in \mathcal{A} \tag{4.30}$$

where $\epsilon_j' = \dfrac{\Delta_p}{t_{p+1} - d_j + p_j}$

Proof. Similar to Lemma 3. □

4.5.3 Third family

We define $\mathcal{S}^j(s) = \{l \in \mathcal{L} : [s, s+p_j] \cap I_l \neq \emptyset\}$ as the set that includes all the indices of those intervals during which j must be in progress if it starts at s ($j \in \mathcal{A}$, $s \in [r_j, d_j - p_j]$). Also, let:

$$\theta_{jl}(s) = \frac{\min(t_{l+1}, s + p_j) - \max(t_l, s)}{\Delta_l}$$

be the fraction of the interval $[t_l, t_{l+1})$ during which activity j is in progress if it starts at s.

Thus, we have the following result

Lemma 5. *The following inequalities*

$$\min_{s \in [r_j, d_j - p_j]} \sum_{l \in \mathcal{S}^j} \theta_{jl}(s) \leq \sum_{l \in \mathcal{I}^j} \frac{x_{jl}}{\Delta_l} \leq \max_{s \in [r_j, d_j - p_j]} \sum_{l \in \mathcal{S}^j} \theta_{jl}(s), \quad \forall j \in \mathcal{A} \tag{4.31}$$

are valid.

Proof. These inequalities trivially hold for any nonpreemptive schedule. □

For each activity $j \in \mathcal{A}$, and each $s \in [r_j, d_j - p_j]$, the computation of $\sum_{l \in \mathcal{S}^j} \theta_{jl}(s)$ requires $O(\rho)$-time. Hence, For each $j \in \mathcal{A}$, the generation of (4.31) requires $O(\upsilon\rho)$-time.

Example 5. *Consider an activity $j \in \mathcal{A}$ with $p_j = 6$, $r_j = 0$, and $d_j = 9$. Also, assume that the time intervals that overlap with $[r_j, d_j)$ are the following: $I_1 = [0, 2)$, $I_2 = [2, 5)$, $I_3 = [5, 6)$, and $I_4 = [6, 9)$. The set of feasible start times is $\{0, 1, 2, 3\}$. We see from Figure 4.2 that:*

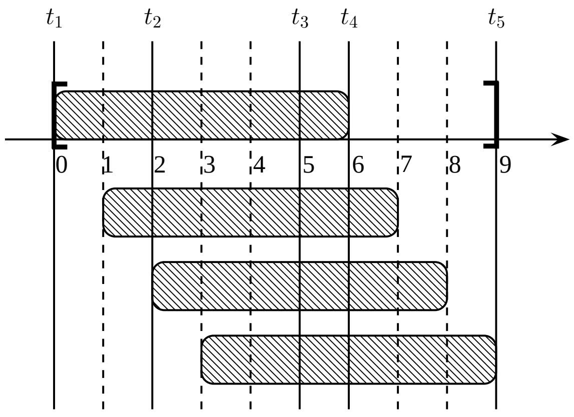


Figure 4.2: Example of nonpreemptive cuts (third family)

- $s = 0 \Rightarrow \sum_{l=1}^{4} \frac{x_{jl}}{\Delta_l} = 1 + 1 + 1 = 3$
- $s = 1 \Rightarrow \sum_{l=1}^{4} \frac{x_{jl}}{\Delta_l} = \frac{1}{2} + 1 + 1 + \frac{1}{3} = \frac{17}{6}$
- $s = 2 \Rightarrow \sum_{l=1}^{4} \frac{x_{jl}}{\Delta_l} = 1 + 1 + \frac{2}{3} = \frac{8}{3}$

- $s = 3 \Rightarrow \sum_{l=1}^{4} \frac{x_{jl}}{\Delta_l} = \frac{2}{3} + 1 + 1 = \frac{8}{3}$

Thus, in any nonpreemptive schedule, we have

$$\frac{8}{3} \leq \sum_{l=1}^{4} \frac{x_{jl}}{\Delta_l} \leq 3$$

4.5.4 Fourth family

Now, consider two intervals I_p and $I_q \in I^j$ $(p < q)$ that satisfy:

$$p_j < t_q - t_{p+1} + 1$$

Thus, activity $j \in \mathcal{A}$ can not be both processed during I_p and I_q. Hence, I_p and I_q are mutually exclusive for activity j. Define Ψ^j as the family of *maximal* sets of intervals (for inclusion) that are mutually exclusive for activity j.

Lemma 6. *A valid constraint is*

$$\sum_{l \in \Psi} \frac{x_{jl}}{\Delta_l} \leq 1, \quad \forall j \in \mathcal{A}, \forall \Psi \in \Psi^j \tag{4.32}$$

Proof. Let Ψ be a maximal set of intervals that are mutually exclusive for activity j. Assume that $\frac{x_{jp}}{\Delta_p} > 0$ for $p \in \Psi$. Then, we have $\sum_{l \in \Psi \setminus \{p\}} \frac{x_{jl}}{\Delta_l} = 0$. □ □

Remark 10. *We can generate constraints (4.32) dynamically. For this, we construct a graph whose nodes represent intervals I_l, $l \in \mathcal{I}^j$ where activity $j \in \mathcal{A}$ can be in execution. To each node, we associate a weight equal to $\frac{\bar{x}_{jl}}{\Delta_l}$ where $\bar{x}_{jl}$, $\forall j \in \mathcal{A}$, $\forall l \in \mathcal{I}^j$, represents the values of the LP solution. Two nodes are connected by an arc if the two corresponding intervals are mutually exclusive. Then, we solve the MWCP by the exact method of Östergård [101]. A violated constraint is detected if the value of the clique is strictly greater than 1. We add this constraint to the LP and we resolve it.*

4.6 Energetic reasoning-based cuts

In this section, we provide three simple, though computationally useful, cuts.

4.6.1 The concept of mandatory parts

Now, we turn our attention to valid constraints that use information dealing with the mandatory part of activities. We recall that the mandatory part of an activity $j \in \mathcal{A}$

during a time interval $[t_1, t_2] \subseteq [0, C]$, hereafter denoted by $p(j, t_1, t_2)$, is the part of j that must be processed within $[t_1, t_2]$. Thus, we have:

$$p(j, t_1, t_2) = \min\left(t_2 - t_1, p_j, \max(0, r_j + p_j - t_1), \max(0, t_2 - d_j + p_j)\right)$$

Using these definitions, we straightforwardly derive the following valid variable bounds:

$$x_{jl} \geq p(j, t_l, t_{l+1}), \quad \forall j \in \mathcal{A}, \forall l \in \mathcal{I}^j \tag{4.33}$$

Clearly, (4.33) dominate (4.6) and can therefore replace these restrictions.

4.6.2 A work-based cut

In the following, we briefly recall the notion of work and we show how it could be used to generate effective cuts.

A simple estimate of the total work $W_k(t_1, t_2)$ over the time-interval $[t_1, t_2]$ and resource k is given by:

$$W_k(t_1, t_2) = \sum_{j \in \mathcal{A}} b_{jk} p(j, t_1, t_2)$$

A key observation is the following:

Proposition 3. *Given an interval I_l ($l \in \mathcal{L}$), a valid constraint is:*

$$\sum_{j \in \mathcal{A}^l} b_{jk} x_{jl} \geq W_k(t_l, t_{l+1}), \quad \forall k \in \mathcal{R}. \tag{4.34}$$

Proof. It suffices to observe that in any feasible schedule, activity j must be processed within I_l during at least $W_{jk}(t_1, t_2)/b_{jk}$ units of time. Thus, we have $b_{jk} x_{jl} \geq W_{jk}(t_l, t_{l+1})$ $\forall j \in \mathcal{A}^l$, $\forall k \in \mathcal{R}$. Hence, the result follows. $\square$

In Section 3.4, we described the so-called Revisited Energetic Reasoning (RER) that aims at deriving a tighter estimate $\tilde{W}_k(t_l, t_{l+1})$ of the total work over a time interval (thus, yielding a stronger cut (4.34)).

Remark 11. *We also implemented an extended variant of (4.33) and (4.34) where further time-intervals $[t_1, t_2]$ (in addition to intervals I_l ($l \in \mathcal{L}$)) are explicitly considered. However, we found that these additional constraints did not impinge on the model performance.*

4.6.3 No idle-time

Finally, a trivial (though useful) valid constraint requires that in any optimal solution, there is at least one activity in progress at each time point:

$$\sum_{j \in \mathcal{A}^l} x_{jl} \geq \Delta_l, \quad \forall l \in \mathcal{L} \tag{4.35}$$

4.7 Experimental results

In this section, we report the results of a computational study that aims at assessing the performance of the proposed cuts. To that aim, we carried out three sets of experiments. In the first set, we investigate the best combination of valid constraints that offers a good trade-off between tightness and efficacy. In the second set, we analyze the performance of our LP-based destructive bound with state-of-the-art lower bounds of Brucker and Knust [23] and Baptiste and Demassey [10], respectively. Finally, in the third set, we compare our lower bound to the constraint programming-based lower bound of Schutt et al. [96].

We used as a testbed the PSPLIB benchmark instances proposed by Kolisch et al. [66]. All the experiments were carried out on a 3.0 GHz Personal Computer with 1 GB RAM. We used CPLEX 11 for solving the LPs.

4.7.1 Performance of the proposed cuts

In this first experiment, we restricted our attention to the 30- and 60-activity instances of PSPLIB. We only considered nontrivial instances for which the best known upper bound is strictly larger than the lower bound LB_{RER} yielded by the RER procedure that is described in Section 3.6. We implemented LB_{RER} as an input for initializing the process (and also for adjusting initial ready times and deadlines).

In this first phase, we analyzed the performance of the following bounds:

- LB_{RER} : the RER-based lower bound
- $LB0$: the destructive lower bound based on Formulation $LP0$ along with the mandatory part constraint (4.33), the no idle-time constraint (4.35) and the precedence-based cut (4.9).
- $LB1$: the destructive lower bound based on the formulation that is obtained after appending, in a preprocessing step, 80 clique constraints (4.23) to the formulation that is used to compute $LB0$.
- $LB2$: the destructive lower bound based on the formulation that is obtained after appending the forbidden activity sets constraints (4.11, 4.12, 4.13 and the dual feasible functions-based constraints 4.27) to the formulation that is used to compute $LB1$.
- $LB3$: the destructive lower bound based on the formulation that is obtained after appending the nonpreemptive constraints (4.28, 4.29, 4.30, 4.31 and 4.32) to the formulation that is used to compute $LB2$. For the sake of efficiency, Constraints (4.32) are restricted to the particular (simple) case of pairwise exclusive intervals.

- $LB4$: the destructive lower bound based on the formulation that is obtained after appending (on the fly) the fourth family of nonpreemptive cuts as described in Remark 10 to the formulation that is used to compute $LB3$.
- $LB5$: the destructive lower bound based on the formulation that is obtained after appending (on the fly) the clique cuts (4.23) to the formulation that is used to compute $LB3$.
- $LBall$: the destructive lower bound based on the formulation that is obtained after both appending (on the fly) the fourth family of nonpreemptive cuts as well as the clique cuts to the formulation that is used to compute $LB3$.

Remark 12. *We performed preliminary experiments that demonstrate that incremental search is faster than dichotomous search. Thus, for the sake of efficacy, all the aforementioned destructive lower bounds are based on an incremental destructive search strategy.*

Tables 4.2-4.3 display a summary of the computational results. For each lower bound, we provide: GAP : the mean percentage deviation with respect to the best known upper bound, $Time$: the mean CPU time (in seconds), $\#(LB = UB)$: number of times where the lower bound is equal to the best known upper bound, $\#(LB = \max LB)$: number of times where the lower bound is maximal, $\#(LB > LB_{RER})$: number of times where the lower bound outperforms the RER-based bound, and TAD : total absolute deviation from LB_{RER}.

	GAP(%)	$Time$(s)	$\#(LB = UB)$	$\#(LB = maxLB)$	$\#(LB > LB_{RER})$	TAD
LB_{RER}	7.29	1.26	0	71	0	0
$LB0$	7.29	1.28	0	71	0	0
$LB1$	5.58	2.50	4	102	37	188
$LB2$	5.57	2.98	4	103	37	189
$LB3$	**5.43**	**3.96**	5	**113**	43	201
$LB4$	5.43	4.86	5	113	43	201
$LB5$	5.42	4.13	5	114	43	202
$LBall$	**5.42**	**5.03**	5	**114**	43	202

Table 4.2: Results of the bounds on 114 non trivial KSD30 instances

	GAP(%)	$Time$(s)	$\#(LB = UB)$	$\#(LB = maxLB)$	$\#(LB > LB_{RER})$	TAD
LB_{RER}	8.06	8.41	0	97	0	0
$LB0$	8.06	8.86	0	97	1	1
$LB1$	7.95	10.50	1	105	10	14
$LB2$	7.95	11.31	1	106	11	15
$LB3$	**7.86**	**20.83**	1	**116**	20	26
$LB4$	7.86	73.65	1	116	20	26
$LB5$	7.85	26.36	1	117	21	27
$LBall$	**7.84**	**83.26**	1	**118**	21	28

Table 4.3: Results of the bounds on 118 non trivial KSD60 instances

We observe that all the new proposed bounds both outperform the initial bound LB_{RER} and the simple LP-based bound $LB0$. Moreover, the results reveal that the new proposed bounds can be roughly classified into two families $\{LB1, LB2\}$ and $\{LB3, LB4, LB5, LBall\}$, where each family includes bounds that exhibit very similar performances. Clearly, bounds of the first family are the fastest, while those that belong to the second family exhibit the best quality.

Interestingly, we observe that the best trade-off between efficiency and effectiveness is achieved by $LB3$. Therefore, in the sequel we shall use this bound as our reference bound and denote it by LB_{Pmtn}.

4.7.2 Comparison with lower bounds of Brucker and Knust [23] and Baptiste and Demassey [10]

In this second experiment, we compare the performance of LB_{Pmtn} with respect to state-of-the-art bounds. To that aim, we consider the full set of benchmark instances of the PSPLIB.

In Table 4.4, we display the results of LB_{Pmtn} as well as those provided by Brucker and Knust's bound (LB_{BK}). Moreover, we included for the set of instances KSD60 the results provided by the bound LB_{BD} that was proposed by Baptiste and Demassey [10] (these authors tested their bound only on KSD60). The column's headings have the same meanings as those of Tables 4.2-4.3. that the reported CPU times of LB_{BK} were obtained on a Sun Ultra 2 workstation 167 MHz and those of LB_{BD} on a HP Omnibook Pentium III running at 720 MHz.

	KSD30				**KSD60**			
	GAP(%)	$Time$(s)	$\#(LB = UB)$	$\#(LB = maxLB)$	GAP(%)	$Time$(s)	$\#(LB = UB)$	$\#(LB = maxLB)$
LB_{Pmtn}	**1.29**	0.9	**371**	480	1.93	3.66	363	400
LB_{BK}	1.50	0.4	318	-	1.85	5	342	409
LB_{BD}	-	-	-	-	**1.35**	86.46	**366**	465
	KSD90				**KSD120**			
LB_{Pmtn}	**1.58**	10.97	**371**	444	**3.41**	96.57	**236**	539
LB_{BK}	1.62	72	353	436	3.62	355 (*)	214	474

(*) Average calculated for 481 instances solved within 1 hour

Table 4.4: Comparison of the bounds on KSD instances

Interestingly, we observe that LB_{Pmtn} exhibits an excellent performance both in terms of average gap as well as the number of times it yields a proven optimal value. Indeed, it consistently exhibits the best performance on all problems sets, but KSD60, where LB_{BD} is the champion bound. Nevertheless, we observe that for this latter problem class, LB_{BD} yields proven optimal values for 76.25% of the instances, while LB_{Pmtn} is only marginally outperformed as it provides proven optimal values for 75.62% of the instances.

Pushing our analysis a step further, we performed a pairwise comparison of the different lower bounds. The results are displayed in Table 4.5.

Table 4.5 provides further evidence of the good performance of LB_{Pmtn}. In particular, we observe that for 120-activity instances LB_{Pmtn} strictly dominates LB_{BK} for 126 instances, while LB_{BK} outperformed LB_{Pmtn} for only 61 instances.

Overall, the computational experiments attest the efficacy of LB_{Pmtn}. This performance is further demonstrated by the fact that LB_{Pmtn} provides a maximal lower bound

	KSD60	KSD90	KSD120
$LB_{Pmtn} > LB_{BK}$	59	44	126
$LB_{Pmtn} = LB_{BK}$	369	400	413
$LB_{Pmtn} < LB_{BK}$	52	36	61
$LB_{Pmtn} > LB_{BD}$	15	-	-
$LB_{Pmtn} = LB_{BD}$	385	-	-
$LB_{Pmtn} < LB_{BD}$	80	-	-

Table 4.5: Comparison between LB_{Pmtn}, LB_{BK} and LB_{BD} on KSD instances

for 1863 instances (91.32%).

4.7.3 Comparison with the approach of Schutt et al. [96]

In this third experiment, we compare the performance of LB_{Pmtn} to the procedures that were proposed by Shutt et al. [96]. In this recently published paper, the authors implemented several exact methods that are based on constraint programming. For all these procedures, the authors set the maximum CPU time to 10 minutes. However, if no proven optimal solution is found within the time limit, then a lower bound is computed using a destructive search procedure that starts from the best known lower bound found on the PSPLIB benchmark. Hence, to make a fair comparison with the approach of Shutt et al. (2011), we used for computing LB_{Pmtn} this same start trial value for all unsolved instances.

Table 4.6 provides a pairwise comparison between LB_{Pmtn} and the lower bound of Shutt et al. [96] (hereafter, denoted by LB_{CP}).

	KSD60	KSD90	KSD120
$LB_{Pmtn} > LB_{CP}$	0	6	47
$LB_{Pmtn} = LB_{CP}$	391	429	453
$LB_{Pmtn} < LB_{CP}$	89	45	100

Table 4.6: Comparison between LB_{Pmtn} and LB_{CP} on KSD instances

Looking at Table 4.6, we see that for most instances (1273 out of 1560) both bounds are equal. On the other hand, although LB_{CP} often outperforms LB_{Pmtn}, we see that LB_{Pmtn} strictly dominates LB_{CP} for 53 instances amongst which we found 48 new improved lower bounds. These new values and the corresponding instances are provided in Appendix A.

As an indication, we compare the execution times of our lower bound LB_{Pmtn} to the exact methods of Shutt et al. (2011). For this, we look to the fastest procedures, on all instance sizes, among those implemented by Schutt et al. [96]. We note however that the execution times correspond to exact methods. The calculation of the lower bounds by Schutt et al. [96] is not reported in the following. The authors used a Computer with an Intel Xeon CPU 2Ghz. The codes were written in Mercury using G12 constraint programming platform.

The best procedure of Schutt et al. [96] on KSD60 has an average execution time of 66.01 seconds against 3.66 seconds for our lower bound. On KSD120 instances, our lower bound shows a processing time of 96.57 seconds against 324.51 seconds on average for the best procedure of Schutt et al. [96].

4.8 Conclusion

In this chapter, we described a new destructive bound for the RCPSP. This bound is based on solving a sequence of relaxed feasibility RCPSP models. Starting from a basic previously suggested LP model, we proposed several original valid inequalities that aim at tightening the model representation. These new inequalities are based on precedence constraints, incompatible activity subsets, dual feasible functions and energetic reasoning. We presented the results of an extensive computational study that was carried out on 2040 benchmark instances with up to 120 activities and that provides evidence that the new proposed lower bound exhibits an excellent performance and often dominates state-of-the-art lower bounds.

Chapter 5

Exact Methods for the Resource Constrained Project Scheduling Problem

5.1 Introduction

Branch-and-Bound algorithms are probably the most widely used solution technique for solving exactly the project scheduling problems. A lot of branching schemes were proposed in the literature. A well known branching scheme is the precedence tree branching scheme (PTBS) which consists in listing all activities in a sequence such that no successor of an activity is sequenced before its predecessor. The most efficient algorithm based on this branching scheme was proposed by Sprecher [98]. Unfortunately, the proposed algorithm as well as the best known branch-and-bound algorithm proposed by Demeulemeester and Herroelen [41] fail to solve 60-activity instances of the PSPLIB benchmark. The first objective of this chapter is to study the impact of the previously proposed lower bounds on a branch-and-bound algorithm based on the PTBS. The second objective consists in proposing new variants of branching schemes derived from the PTBS.

The remainder of this chapter is organized as follows. In Section 5.2, we begin by reviewing the precedence tree branching scheme and the dominance rules allowing the reduction of the search tree. In section 5.3, we present two variants of the PTBS. Finally, the results of our computational experiments are presented in Section 5.4.

5.2 The precedence tree branching scheme

5.2.1 Description of the scheme

In the precedence tree branching scheme, the root node N_0 is given by the dummy start activity 0 and the leaves correspond to copies of the dummy finish activity $n+1$. The descendants of a node N_j within the precedence tree are built by the activities that are eligible after scheduling the activities on the path leading from the root node N_0 to node N_j. Thus, an activity is called eligible if all its predecessors are scheduled. The algorithm can be stated as follows. Only one activity is scheduled per node of the branch-and-bound. The start time ST_{A_j} of activity A_j, considered for scheduling at level j, is the lowest feasible start time which respects the following conditions:

1. The start time ST_{A_j} is greater than the start time of the activity most recently scheduled,
2. The start time ST_{A_j} is lower than the latest start time (equal to $d_{A_j} - p_{A_j}$) of A_j,
3. The scheduling of A_j does not violate the precedence and resource constraints.

Note that employing this strategy reduces substantially the search tree [97].

If scheduling of the current activity is not feasible then backtracking is performed. The next untested eligible activity is then selected. If there is no untested eligible activity left then backtracking to the previous level is performed. Thus, the next eligible activity is chosen.

Clearly, the ordering of the eligible set, i.e., the decision which activity to select when branching, has an influence on the solution time. However, for the rest of the chapter, we assume, without loss of generality, that the eligible sets are arranged with respect to activity numbers. Some of the priority rules allowing to relabel the activities before the enumeration will be tested in Section 5.4.

In the sequel, the various exact procedures will be illustrated on the example instance of Figure 5.1.

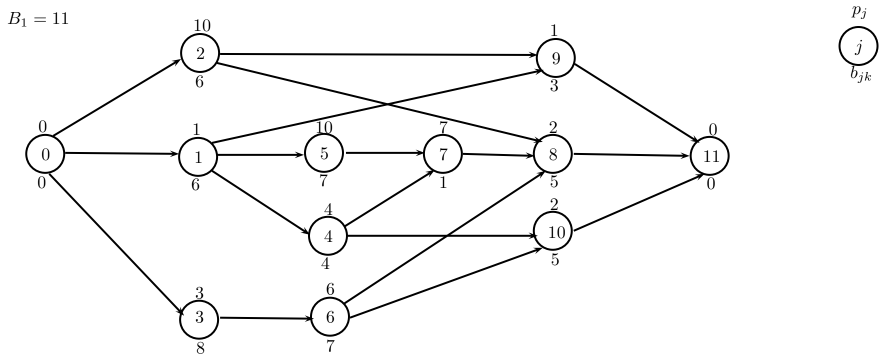


Figure 5.1: Exact methods instance example

A sample of the search tree produced by the branch-and-bound algorithm based on the PTBS is presented in Figure 5.2.

We can see from the figure that in level 1, we have three nodes corresponding to the three eligible activities 1, 2 and 3 (since these activities have no predecessors). In level 2, we see that node 1 has 4 children corresponding to the scheduling of eligible activities 2, 3, 4 and 5. These activities became eligible as their predecessor (activity 1) was scheduled. All these activities cannot begin before time instant 1 (corresponding to

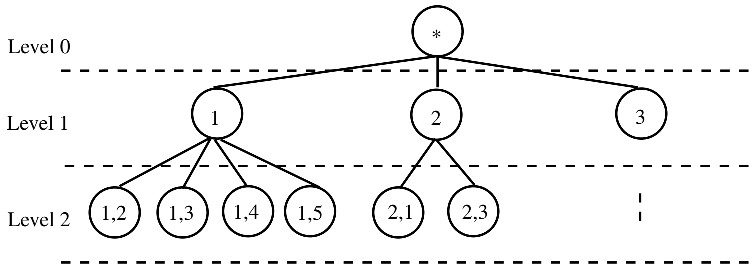


Figure 5.2: Search tree produced by the PTBS

the completion time of activity 1). Indeed, activities 2 and 3 cannot be scheduled at time instant 0 due to resource conflicts. In addition, activities 4 and 5 have to be scheduled after the completion time of activity 1 due to the precedence constraint. For node 2, it has only two children because the successors of activity 2 (activities 8 and 9) still have unscheduled predecessors.
Following this strategy, all possible combinations will be explored. Then, it will be necessary to add dominance rules in order to reduce the search space and speed up the convergence of the branch-and-bound.

5.2.2 Dominance rules

In the sequel, we use the following definitions:

- $\mathcal{CS}_j$: the set of currently scheduled activities on level j,
- $\mathcal{NS}_j$: the set of currently unscheduled activities on level j,
- $\mathcal{EL}_j$: the set of currently eligible activities on level j,
- $Seq_j = [A_1, A_2, \ldots, A_j]$: the currently considered sequence on level j where A_1 is the 1^{st} scheduled activity and A_j is the j^{th} scheduled activity,
- ST_{A_j}: Start time of activity A_j,
- CT_{A_j}: Completion time of activity A_j.

5.2.2.1 The extended and simplified single enumeration rule

This rule was introduced by Sprecher [98]. It extends the single enumeration rule presented by Sprecher and Drexl [99]. The rule excludes multiple enumeration of the same partial schedule induced by different sequences. It is formulated as follows:

Proposition 4. *Let $Seq_{j+1} = [A_1, \ldots, A_i, A_{j+1}]$ be the currently considered sequence. If $A_{j+1} < A_j$ and $ST_{A_{j+1}} = ST_{A_j}$, then the current sequence is dominated by the previously evaluated sequence $Seq'_{j+1} = [A_1, \ldots, A_{j+1}, A_j]$.*

5.2.2.2 Local and global left-shift rule

The following dominance rules are based on the concept of left-shift of activities [100]. A left-shift consists in reducing the start time of an activity while preserving the start times of the remaining activities. A one-period left-shift derives a feasible schedule through reducing the start time of one activity by one time period. Based on this concept, we can distinguish two types of left-shifts: local left-shifts and global ones. A local left-shift is a left-shift which can be obtained by a series of one-period left-shifts. A feasible schedule for which no further local left-shifts can be performed is called a *semi-active schedule*. A global left-shift is a left-shift operation which cannot be obtained by a series of one-period left-shift. Feasible schedules in which no local or global left-shifts can be performed are called *active schedules*. It has been proven by Sprecher et al. [100] that for every RCPSP instance at least one optimal schedule exists that is active.

Formally, these dominance rules are presented as the following

- *Local left-shift*: Let $Seq_j = [A_1, \dots, A_j]$ be the currently considered sequence to be continued with activity A_{j+1}. If the start time $ST_{A_{j+1}}$ can be reduced to $\overline{ST}_{A_{j+1}} = ST_{A_{j+1}} - 1$ without violating the precedence and resource constraints, then the current selection can be skipped.

- *Global left-shift*: Let $Seq_j = [A_1, \dots, A_j]$ be the currently considered sequence to be continued with activity A_{j+1}. If the start time $ST_{A_{j+1}}$ can be reduced to $\overline{ST}_{A_{j+1}} \leq ST_{A_j} - p_{A_j}$ without violating the precedence and resource constraints, then all the continuations of Seq_j can be skipped.

5.2.2.3 Non-optimality rule

The non-optimality rule was also introduced by Sprecher [98]. It is formulated as follows. Let $Seq_j = [A_1, \dots, A_j]$ be the currently considered sequence to be continued with activity A_{j+1}. If (a) $ST_{A_{j+1}} = CT^{max}(Seq_j)$ where $CT^{max}(Seq_j) = \max_{i=1,\dots,j} CT_{A_i}$, (b) $\Delta_{A_{max}}$ periods of activity A_{max} inducing $CT^{max}(Seq_j)$ can be feasibly left-shifted, and (c) the difference Δ_{max} between $CT^{max}(Seq_j)$ and the second largest completion time $CT^{\overline{max}}(Seq_j)$ of the activities already scheduled is larger than the non-left-shiftable portion of activity A_{max}, i.e, $\Delta_{max} = CT^{max}(Seq_j) - CT^{\overline{max}}(Seq_j) > p_{A_{max}} - \Delta_{A_{max}}$, then a continuation of $Seq_{j+1} = [A_1, \dots, A_j, A_{j+1}]$ cannot be makespan minimal.

Remark 13. *The maximal left-shiftable portion $\Delta_{A_{max}}$ can be obtained as a by-product of the global left-shift rule.*

5.2.2.4 Cutset dominance rule

This dominance rule was initially introduced by Demeulemeester and Herroelen [40]. It is based on the notion of *Cutset*. A cutset $\mathcal{C}_t$ at time instant t is defined as the set of eligible activities (for which all predecessors are in the set of scheduled activities). It is clear that a cutset does not depend on a time instant but, only, on the set of scheduled activities. Thus, there is a correspondence between a cutset and the set of scheduled activities $\mathcal{PS}_t$. The cutset dominance rule is stated as follows.

Proposition 5. *Consider a cutset $\mathcal{C}_t$ at time t which contains the same activities as a cutset $\mathcal{C}_{t'}$, which was previously saved during the search tree. If time t' was not greater than time t and if all activities in progress at time t' did not finish later than the maximum of t and the finish time of the corresponding activities in $\mathcal{PS}_t$ then the partial schedule $\mathcal{PS}_t$ is dominated.*

In our implementation, the time instants t correspond to the start times of the currently considered activity to be scheduled.

5.3 Forward-Backward branching schemes

It is worth noting that the RCPSP is symmetric, i.e., the original instance and the reverted one (obtained by inversing the precedence constraints) have the same makespan. Thus, an interesting idea to be investigated is the possibility of placing alternatively the activities at the beginning and the end of the schedule. This can result to the reduction of the time-windows (by reducing the release dates and due dates) of activities so infeasibilities may be detected more rapidly. The main idea consists in:

- scheduling the *forward activities* (i.e, placed at the beginning of the schedule) at their shortest possible start time,
- scheduling the *backward activities* (i.e, placed at the end of the schedule) at their longest possible finish time,
- respecting the precedence and the resource constraints.

In the following, this method will be referred as the *Forward-Backward Branching Scheme* (FBBS).

Example 6. *Given an upper bound UB equal to 35, a schedule (for the instance example of Figure 5.1) obtained by the FBBS is presented in Figure 5.3.*

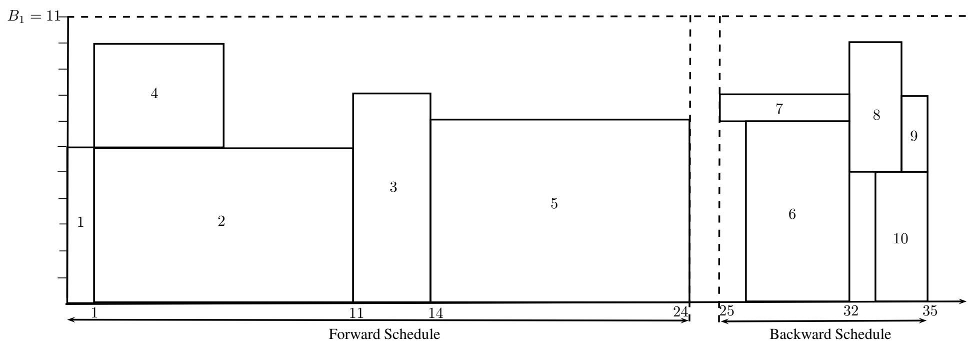


Figure 5.3: A schedule obtained by the forward-backward scheme

Although the idea of FBBS is interesting in theory, the method described previously has problem. It consists in the fact that it is difficult to re-construct a feasible solution respecting the chronological order on the start times from the obtained solution (see Figure 5.3). Indeed, if the backward activities are simply left shifted, in order to fix the two schedules, then the optimal schedule can be skipped. This is because it is not proven that for each schedule respecting the chronological order on the finish times there exists a schedule respecting the chronological order on the start times.

In the following, we propose two variants of the FBBS in order to get around this problem.

5.3.1 Implicit enumerative search algorithm

The first method consists in a dichotomic procedure. At each iteration, we fix a value C and we try to find a schedule which fits into C. For this, the *Implicit Enumerative Search Algorithm* (IESA) schedules, alternatively, the activities at their shortest possible start time (greatest possible finish time) such a way to respect the precedence and resource constraints, i.e., if we schedule , in the order, activities $1, 2, \ldots, 2q$, then we have:

$$ST_1 \leq ST_3 \leq \ldots ST_{2q-1}$$
and
$$C_2 \geq C_4 \geq \ldots C_{2q}$$

where $ST_1 = 0$ and $C_2 = C$.

If the IESA reaches a leaf then C is a valid upper bound. Otherwise, the IESA explores all the tree without reaching any leaf. In that case, $C + 1$ is a lower bound.

The procedure is repeated until the optimal value is reached. This procedure is summarized in Algorithm 3.

Algorithm 3 IESA-based branch-and-bound

```
1:  $C \leftarrow \left\lfloor \frac{LB + UB}{2} \right\rfloor$
2:  Call the IESA procedure with value $C$
3:  **if** IESA returns Yes **then**
4:      $UB \leftarrow C$
5:  **else**
6:      $LB \leftarrow C + 1$
7:  **end if**
8:  **if** $UB = LB$ **then**
9:      Stop the procedure
10: **else**
11:     Go to line 2
12: **end if**
```

5.3.2 Bloc-based branching scheme

The key observation of this method is that a schedule can be decomposed into a set of successive blocs. The blocs are constructed such a way that all the activities of a bloc cannot start earlier than the completion time of its preceding bloc (where the completion time of a bloc is the maximum of the completion times of the activities of that bloc) due to resource or precedence constraints. Figure 5.4 presents an optimal schedule and its decomposition into blocs. We remark from this figure that from time instant 0 to time instant 1, only activity 1 is scheduled. In this time-interval, the eligible activities are 2 and 3. Due to resource constraints, these activities cannot be scheduled simultaneously with activity 1. Thus, the first bloc is constituted only by activity 1. This reasoning is applied over the schedule in order to decompose it into successive blocs.

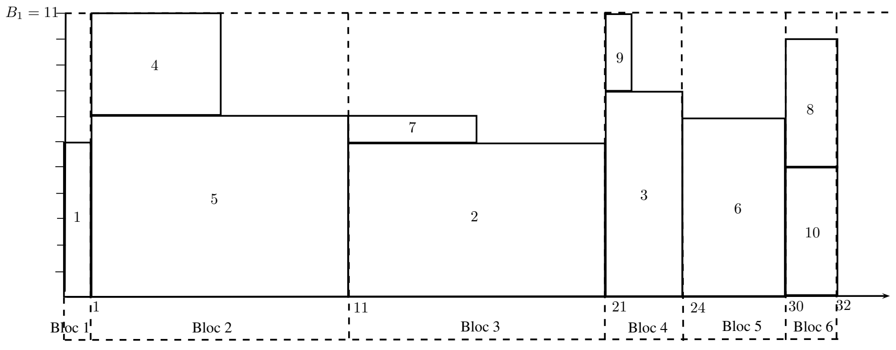


Figure 5.4: Decomposition of a schedule into blocs

A first interesting remark consists in the fact that for each bloc respecting the chronological order on the start times corresponds a bloc (constituted by the same activities) respecting the chronological order on the finish times having the same makespan. This statement can be used for the FBBS. Indeed, the branching scheme is modified in order to begin with placing activities, at the beginning of the schedule, until a bloc is detected. The placement of the activities is then changed to the end of the schedule. We return at placing the activities at the beginning only when a bloc is detected in the backward schedule. This procedure is reiterated until all activities are scheduled. Let D denotes the difference between the start time of the last bloc of the backward schedule and the completion time of the last bloc placed in the forward schedule (see Figure 5.5). The new upper bound is then the difference between the value of the old upper bound and D.
Figure 5.5 presents an optimal schedule obtained by this branching scheme starting with an upper bound equal to 35.

We see from Figure 5.5 that D is equal to 3. Thus, the new upper bound is equal to 35-3=32 (which corresponds to the optimal value).

In the following, this method will be referred as the *Bloc-based Branching Scheme* (BBS).

A set of remarks can be stated concerning the BBS:

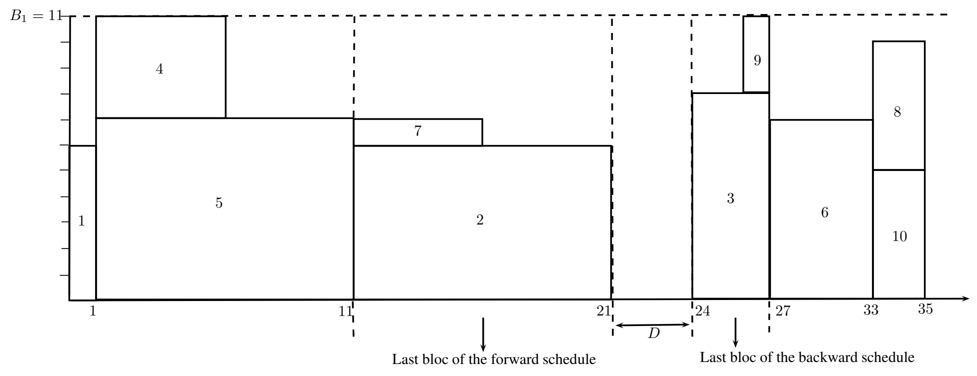


Figure 5.5: Optimal schedule obtained by the bloc-based scheme

1. In the worst case, we cannot detect a bloc. In that case, the BBS corresponds to the original precedence tree branching scheme.
2. The BBS can be interesting for instances with constrained resources. Indeed, for this type of instances blocs can be easily detected.
3. If at a level i, we have a forward schedule of makespan C_f and a backward schedule of makespan C_b, then a valid lower bound is $C_f + C_b + LB(\mathcal{NS}_i)$ where $LB(\mathcal{NS}_i)$ is a lower bound on the instance made up with activities of $\mathcal{NS}_i$.

5.3.3 Adaptation of the dominance rules

In this section, we show how to adapt the dominance rules presented in Section 5.2.2 for the case of the backward schedule.

5.3.3.1 Backward single enumeration rule

Let $Seq_{j+1} = [A_1, \ldots, A_i, A_{j+1}]$ be the currently considered sequence. If $A_{j+1} > A_j$ and $CT_{A_{j+1}} = CT_{A_j}$, then the current sequence is dominated by the previously evaluated sequence $Seq'_{j+1} = [A_1, \ldots, A_{j+1}, A_j]$.

5.3.3.2 Backward local left-shift rule

Let $Seq_j = [A_1, \ldots, A_j]$ be the currently considered sequence to be continued with activity A_{j+1}. If the completion time $CT_{A_{j+1}}$ can be reduced to $\overline{CT}_{A_{j+1}} = CT_{A_{j+1}} + 1$ without violating the precedence and resource constraints, then the current selection can be skipped.

5.3.3.3 Backward global left-shift rule

Let $Seq_j = [A_1, \ldots, A_j]$ be the currently considered sequence to be continued with activity A_{j+1}. If the completion time $CT_{A_{j+1}}$ can be reduced to $\overline{CT}_{A_{j+1}} \geq CT_{A_j} + p_{A_j}$

without violating the precedence and resource constraints, then all the continuations of Seq_j can be skipped.

5.3.3.4 Backward non-optimality rule

If (a) $CT_{A_{j+1}} - p_{A_{j+1}} = ST^{min}(Seq_j)$ where $ST^{min}(Seq_j) = \min_{i=1,\dots,j} ST_{A_i}$, (b) $\Delta_{A_{min}}$ periods of activity A_{min} inducing $ST^{min}(Seq_j)$ can be feasibly left-shifted, and (c) the difference Δ_{max} between $ST^{min}(Seq_j)$ and the second smallest start time $\overline{ST^{min}}(Seq_j)$ of the activities already scheduled is larger than the non-left-shiftable portion of activity A_{min}, i.e, $\Delta_{max} = ST^{min}(Seq_j) - \overline{ST^{min}}(Seq_j) > p_{A_{min}} - \Delta_{A_{min}}$, then a continuation of $Seq_{j+1} = [A_1, \dots, A_j, A_{j+1}]$ cannot be makespan minimal.

5.3.3.5 Backward cutset dominance rule

The cutset dominance rule is not changed for the backward schedule. We simply change the finish times by the start times. However, there is a problem with the application of the rule. Indeed, it can result in eliminating some partial schedules which lead to the optimal solution. To illustrate this point, we suppose that we use the bloc-based branching scheme and we apply it to the example of Figure 5.1. In a path, we schedule bloc {1} then bloc {10;9;8} and finally we return to the forward schedule. In another path, bloc {1} is scheduled then bloc {10;8}. When we return to the forward schedule and apply the cutset dominance rule, we find the same cutset as in the first path. Thus, the application of the rule leads to the elimination of the current node. This results in the elimination of all the schedules with bloc {10;8} at the end of the schedule. Nevertheless, we know from Figure 5.4 that, in the optimal schedule, activities 10 and 8 must be scheduled simultaneously at the end. To overcome this problem, we must explore the two branches of the tree, i.e., the cutset dominance rule do not eliminate a schedule which have a same cutset as a previously stored one. Unfortunately, this leads to a weaker version of the rule.

5.4 Experimental results

To assess the performance of the proposed branch-and-bound algorithms, we considered the set of benchmark instances proposed by Kolisch et al. [66] (see Section 2.6).

All the experiments were conducted on a personal computer Core i5 2.67 Ghz with 4 GB of RAM and running under Windows 7. All the algorithms were coded in C language. We note that all the branch-and-bound algorithms are based on the depth first strategy. The starting upper bound is a deviation of the optimal by 10%.

A pseudo-code of the branch-and-bound procedure based on the precedence tree branching scheme is presented in Algorithm 4.

Algorithm 4 Branch-and-bound procedure based on the PTBS

```
 1: Compute $UB$
 2: Compute $LB$
 3: if $LB = UB$ then
 4:   $UB$ is the optimal solution
 5: else
 6:   Put in the stack the successors of the dummy start activity
 7:   while The stack is not empty do
 8:     Unstack an activity
 9:     Determine the start time of the current activity
10:     if The current node is a leaf then
11:       Compute $NewUB$
12:       if $NewUB = LB$ then
13:         $NewUB$ is the optimal solution
14:         Stop the procedure
15:       else
16:         if $NewUB < UB$ then
17:           $UB \leftarrow NewUB$
18:           Compute the release dates and due dates of the activities with $UB - 1$
19:         end if
20:       end if
21:     else
22:       Answer $\leftarrow$ DominanceRules()
23:       if Answer=Yes then
24:         Kill the current node
25:         Go to line 8
26:       end if
27:       Update the resources profiles
28:       Fix the release dates and due dates of the current activity
29:       Propagation to the successors of the current activity
30:       Compute $NewLB$
31:       if $NewLB \geq UB$ then
32:         Kill the current node
33:         Go to line 8
34:       end if
35:     end if
36:   end while
37: end if
```

5.4.1 Impact of the priority rules

We conducted a first set of experiments in order to assess the impact of the priority rules allowing to relabel the activities before the enumeration. The following priority rules were tested:

- BB_1: arrangement with respect to activity numbers.
- BB_2: arrangement with respect to release dates. Tie break with due dates.
- BB_3: arrangement with respect to release dates. Tie break with latest possible start time of the activities.
- BB_4: arrangement with respect to release dates. Tie break with activity numbers.

Table 5.1 displays a summary of the computational results that were obtained on the 30-activity set of instances. For each branch-and-bound, we provide: NbS: the number of solved instances within the time limit of 1800 seconds, and $MTime$: the mean CPU time (in seconds).

	NbS	$MTime$(s)
BB_1	480	3.92
BB_2	480	3.68
BB_3	480	3.86
BB_4	480	**3.08**

Table 5.1: Impact of the priority rules

We see from Table 5.1 that the best priority rule is the arrangement with respect to release dates and tie break with activity numbers. Thus, we will use this priority rule for the next set of experiments.

5.4.2 Impact of the energetic-based lower bound

To assess the impact of the energetic-based lower bound on the branch-and-algorithm, we tested the following procedures:

- BB_{Ener1}: Branch-and-bound with enhanced energetic-based lower bound (simple feasibility tests) and adjustment procedure.
- BB_{Ener2}: Branch-and-bound with enhanced energetic-based lower bound without adjustment procedure.
- BB_{DFF1}: Branch-and-bound with enhanced energetic-based lower bound DFF^* (see Section 3.5) and adjustment procedure.
- BB_{DFF2}: Branch-and-bound with enhanced energetic-based lower bound DFF^* without adjustment procedure.

	NbS	$MTime$(s)
BB_{Ener1}	480	5.92
BB_{Ener2}	480	**5.22**
BB_{DFF1}	478	8.83
BB_{DFF2}	479	9.23

Table 5.2: Impact of the energetic-based lower bound

We see from Table 5.2 that the procedures based on the enhanced energetic-based lower bound can solve all the 480 instances within a time limit of 1800 seconds. Moreover, BB_{Ener2} is the fastest one among the 4 branch-and-bound algorithms. Nevertheless, this procedure is slower than the BB_4 procedure (see Table 5.1). Thus, the enhanced energetic-based lower bound cannot speed up the branch-and-bound algorithm compared to the critical sequence bound.

5.4.3 Impact of the forward-backward schemes

To assess the effect of the FBBS, we tested the following two procedures.

- BBS_1: Branch-and-algorithm with BBS and critical sequence lower bound.
- BBS_2: Branch-and-algorithm with BBS and enhanced energetic-based lower bound without adjustments.

It is worth noting that preliminary results showed that the procedures based on the IESA are slower than the procedures based on BBS.

Table 5.3 shows the results of the two procedures on the first 160 instances. We see that the procedures are not able to solve all the instances.

	NbS	$MTime$(s)
BBS_1	154	54.49
BBS_2	149	30.78

Table 5.3: Impact of the FBBS

5.5 Conclusion

In this chapter, we proposed new branch-and-bound algorithms in order to solve the RCPSP. First, we adapted the energetic reasoning-based lower bounds developed in Chapter 3 to the well known precedence tree branching scheme. Secondly, we proposed two new branching schemes which construct the schedule, alternatively, at the beginning and the end. Unfortunately, computational results showed that we were not able to improve the results of the literature and to close open instances in the PSPLIB benchmark. A more detailed analysis of the dominance rules must be conducted in order to enhance the efficiency of the proposed branching schemes.

Conclusion

The main contribution of this thesis is the development of new lower bounds as well as exact methods for the Resource Constrained Project Scheduling Problem.

First, in Chapter 3, we focused our attention on the development of lower bounds based on effective improvements of the well-known concept of Energetic Reasoning. Moreover, the concept of Dual Feasible Functions was used in order to derive tighter lower bounds and to enable more effective adjustment of release dates and due dates. Furthermore, we tested a new shaving procedure that enhanced the quality of the lower bounds. Extensive numerical experiments showed the effectiveness of the new proposed lower bounds that often outperform the best bounds from the literature.

At a second stage, in Chapter 4, the previous developed lower bounds were used as an input for a destructive bound that is derived by making feasibility tests in order to detect an infeasibility. If an infeasibility occurs, then the value of the lower bound is incremented and the checking procedure is restarted. The lower bounds tested at this stage are based on LP relaxations of the RCPSP derived from the formulation of Carlier and Néron [29], by partitioning the time horizon into time-windows. We proposed several cuts based on precedence, nonpreemptipn and incompatible activity sets in order to enhance the mathematical formulation. Computational results showed that this technique was able to improve the value of the lower bounds for the instances of the PSPLIB benchmark [66].

Finally, in Chapter 5, we proposed several branch-and-bound procedures to solve exactly the RCPSP. As a first step, we adapted the aforementioned lower bounds for the precedence tree branching scheme-based branch-and-bound. Regrettably, we have not been able to improve the best results of the literature with this method. As a second step, we proposed two new branching schemes in which the solution is built by adding activities alternatively at the beginning and the end of the schedule.

Future research effort needs to be focused on the design of dominance rules for solving exactly the problem. It is clear that the performance of the best branch-and-bound developed by Demeulemeester and Herroelen [41] is due to the Cutset dominance rule. A deeper investigation of this rule and its adaptation to the Bloc-based branching scheme can lead to a stronger version of the rule than the one proposed in Chapter 5 and therefore can strengthen the efficiency of the proposed branch-and-bound. Another worthy issue for future investigation, is to develop a set of heuristic procedures based on mathematical formulations of the RCPSP. The main idea consists in constructing a feasible solution from the preemptive solution given by the mathematical formulation developed in Chapter 4. As a first step, priority rules based on the values of the variables given by the solution can be constructed and the serial and/or parallel scheduling schemes applied. After that, these solutions can be improved by detecting the blocs of a schedule and solves exactly each bloc separately. Finally, an extension of the work conducted throughout this thesis consists in adapting the lower bounds and exact methods for variants and generalizations of the RCPSP such as the multi-mode and multi-skill project scheduling problems.

Appendices

Appendix A

New Lower Bounds

Tables A.1-A.2 list all the improved lower bounds for the KSD instances of the PSPLIB benchmark.

Instance	**New LB**
5_3	83
5_5	108
9_6	111
9_8	110
9_10	103
41_10	142

Table A.1: New lower bounds for KSD90 instances

Instance	New LB
6_5	116
7_4	104
7_5	124
7_7	113
7_9	85
8_9	88
12_3	131
12_7	116
12_8	113
12_9	101
13_5	90
13_10	87
14_2	90
14_8	109
18_8	102
19_4	102
19_9	88
26_1	155
26_3	156
26_6	168
27_3	140
27_4	104
27_6	131
27_9	120
28_1	105
31_1	179
31_3	157
32_4	126
32_7	118
32_8	131
33_3	101
34_5	101
37_1	138
37_6	154
39_2	105
46_8	164
47_6	128
47_7	112
47_10	127
48_4	121
53_1	137
53_8	134

Table A.2: New lower bounds for KSD120 instances

Bibliography

[1] A. Agarwal, S. Colak, and S. Erenguc, *A neurogenetic approach for the resource-constrained project scheduling problem*, Computers & Operations Research **38** (2011), 44–50.

[2] J. Alcaraz, C. Maroto, and R. Ruiz, *Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms*, Journal of the Operational Research Society **54** (2003), 614–626.

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

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

[5] C. Artigues, S. Demassey, and E. Néron (eds.), *Resource-constrained project scheduling : Models, algorithms, extensions and applications*, Wiley, 2008.

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

[7] T. Baar, P. Brucker, and S. Knust, *Tabu-search algorithms and lower bounds for the resource constrained project scheduling problem*, Meta-heuristics: Advances and trends in local search paradigms for optimization, pp. 1–8, Kluwer Academic Publishers, 1998.

[8] F. Ballestín, V. Valls, and S. Quintanilla, *Perspectives in modern project scheduling*, ch. Due dates and RCPSP, pp. 131–163, Springer, 2006.

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

[10] Ph. Baptiste and S. Demassey, *Tight lp bounds for resource constrained project scheduling*, OR Spectrum **26** (2004), 251–262.

[11] Ph. Baptiste and C. Le Pape, *Constraints propagation and decomposition techniques for highly disjunctive and highly cumulative project scheduling problems*, Lecture Notes in Computer Science **1330** (1997), 375–389.

[12] Ph. Baptiste, C. Le Pape, and W. Nuijten, *Satisfiability tests and time-bound adjustments for cumulative scheduling problems*, Annals of Operations Research **92** (1999), 305–333.

[13] M. Bartusch, R. Mohring, and F. Radermacher, *Scheduling project networks with resource constraints and time windows*, Annals of Operations Research **16** (1988), 201–240.

[14] O. Bellenguez and E. Néron, *Lower bounds for the multi-skill project scheduling problem with hierarchical levels of skills*, Practice and Theory of Automated Timetabling (PATAT'04) (Pittsburgh, PA, USA), 2004, pp. 429–432.

[15] ______, *Methods for solving the multi-skill project scheduling problem*, 9th International Workshop on Project Management and Scheduling (PMS'04) (Nancy, France), 2004, pp. 66–69.

[16] ______, *A branch-and-bound method for solving multi-skill project scheduling problem*, RAIRO Operations Research **41** (2007), 155–170.

[17] L. Bianco and M. Caramia, *A new lower bound for the resource-constrained project scheduling problem with generalized precedence relations*, Computers & Operations Research **38** (2011), 14–20.

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

[19] F. Boctor, *A new and efficient heuristic for scheduling problems with resource restrictions and multiple execution modes*, European Journal of Operational Research **90** (1996), 349–361.

[20] K. Bouleimen and H. Lecocq, *A new efficient simulated annealing algorithm for the resource constrained project scheduling problem and its multiple modes version*, European Journal of Operational Research **149** (2003), 268–281.

[21] G. Brooks and C. White, *An algorithm for finding optimal or near optimal solutions to the production scheduling problem*, Journal of Industrial Engineering **16** (1965), 34–40.

[22] P. Brucker and S. Knust, *Lower bounds for resource-constrained project scheduling problems*, European Journal of Operational Research **149** (1998), 302–313.

[23] ______, *A linear programming and constraint propagation-based lower bound for the RCPSP*, European Journal of Operational Research **127** (2000), 355–362.

[24] P. Brucker and S. Knust (eds.), *Complex scheduling*, Springer, 2006.

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

[26] J. Carlier, F. Clautiaux, and A. Moukrim, *New reduction procedures and lower bounds for the two-dimensional bin packing problem with fixed orientation*, Computers & Operations Research **34** (2007), 2223–2250.

[27] J. Carlier and B. Latapie, *Une méthode arborescente pour résoudre les problèmes cumulatifs*, Recherche Opérationelle **25** (1991), 311–340.

[28] J. Carlier and E. Néron, *A new lp-based lower bound for the cumulative scheduling problem*, European Journal of Operational Research **127** (2000), no. 2, 363–382.

[29] ______, *On linear lower bounds for the resource constrained project scheduling problem*, European Journal of Operational Research **149** (2003), 314–324.

[30] ______, *Computing redundant resources for the resource constrained project scheduling problem*, European Journal of Operational Research **176** (2007), 1452–1463.

[31] J. Carlier and E. Pinson, *Adjustment of heads and tails for the job-shop problem*, European Journal of Operational Research **78** (1994), 146–161.

[32] J. Carruthers and A. Battersby, *Advances in critical path methods*, Operational Research Quarterly **17** (1966), no. 4, 359–380.

[33] Y. Caseau and F. Laburthe, *Cumulative scheduling with task intervals*, Proceedings of the Joint International Conference and Symposium on Logic Programming (MIT Press) (M. Maher, ed.), 1996, pp. 363–377.

[34] A. Cesta, A. Oddi, and S. Smith, *A constraint-based method for project scheduling with time windows*, Journal of Heuristics **8** (2002), 109–136.

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

[36] J. Coelho and M. Vanhoucke, *Multi-mode resource-constrained project scheduling using rcpsp and sat solvers*, European Journal of Operational Research (2011).

[37] D. Debels, B. De Reyck, R. Leus, and M. Vanhoucke, *A hybrid scatter search/electromagnetism meta-heuristic for project scheduling*, European Journal of Operational Research **169** (2006), 638–653.

[38] S. Demassey, C. Artigues, Ph. Baptiste, and Ph. Michelon, *Lagrangean relaxation-based lower bounds for the rcpsp*, 9th international workshop on project management and scheduling (Nancy, France), 2004.

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

[40] E. Demeulemeester and W. Herroelen, *A branch-and-bound procedure for the multiple resource-constrained project scheduling problem*, Management Science **38** (1992), no. 12, 1803–1818.

[41] ______, *New benchmark results for the resource-constrained project scheduling problem*, Management Science **43** (1997), no. 11, 1485–1492.

[42] ______, *Project scheduling: a research handbook*, ISOR, vol. 49, Kluwer Academic Publisher, 2002.

[43] U. Dorndorf, E. Pesch, and T. Phan-Huy, *Constraint propagation techniques for the disjunctive scheduling problem*, Artificial Intelligence **122** (2000), 189–240.

[44] A. Drexl and F. Salewski, *Distribution requirements and compactness constraints in school timetabling*, European Journal of Operational Research **102** (1997), 193–214.

[45] J. Erschler, P. Lopez, and C. Thuriot, *Raisonnement temporel sous contraintes de ressources et problèmes d'ordonnancement*, Revue d'Intelligence Artificielle **5** (1991), 7–32.

[46] C. Fang and L. Wang, *An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem*, Computers & Operations Research **39** (2012), 890–901.

[47] S.P. Fekete and J. Schepers, *New classes of lower bounds for bin-packing problems*, Lecture Notes in Computer Science **1412** (1998), 257–270.

[48] A. Fest, R. Möhring, F. Stork, and M. Uetz, *Resource constrained project scheduling with time windows: A branching scheme based on dynamic release dates*, Tech. report, Technische Universität Berlin, 1998.

[49] M. Fisher, *Optimal solution of scheduling problems using lagrange multipliers: Part I*, Operations Research **21** (1973), 1114–1127.

[50] M. Haouari and A. Gharbi, *Fast lifting procedures for the bin packing problem*, Discrete Optimization (2005), 201–218.

[51] M. Haouari, A. Kooli, and E. Néron, *Enhanced energetic reasoning-based lower bounds for the resource constrained project scheduling problem*, Computers & Operations Research **39** (2012), 1187–1194.

[52] M. Haouari, A. Kooli, E. Néron, and J. Carlier, *A preemptive bound for the resource constrained project scheduling problem*, Journal of scheduling (Submitted).

[53] S. Hartmann, *A competitive genetic algorithm for resource constrained project scheduling*, Naval Research Logistics **45** (1998), 733–750.

[54] ______, *Project scheduling with multiple modes: A genetic algorithm*, Annals of Operations Research **102** (2001), 111–135.

[55] ______, *A self-adapting genetic algorithm for project scheduling under resource constraints*, Naval Research Logistics **49** (2002), 433–448.

[56] S. Hartmann and D. Briskorn, *A survey of variants and extensions of the resource-constrained project scheduling problem*, European Journal of Operational Research **207** (2010), 1–14.

[57] R. Heilmann and C. Schwindt, *Lower bounds for RCPSP/max*, Tech. report, Universität Karlsruhe, 1997.

[58] L. Hidri, A. Gharbi, and M. Haouari, *Energetic reasoning revisited: application to parallel machine scheduling*, Journal of Scheduling **11** (2008), 239–252.

[59] B. Jarboui, N. Damak, P. Siarry, and A. Rebai, *A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems*, Applied Mathematics and Computation **195** (2008), 299–308.

[60] J. Jozefowska and J. Weglarz (eds.), *Perspectives in modern project scheduling*, W.H. Freeman & Co, 2006.

[61] J. Kelley, *The critical-path method: Resources planning and scheduling*, Industrial Scheduling, pp. 347–365, Prentice Hall, Englewood Cliffs, 1965.

[62] K. Kim, M. Gen, and G. Yamazaki, *Hybrid genetic algorithm with fuzzy logic for resource-constrained project scheduling*, Applied Soft Computing **2/3F** (2003), 174–188.

[63] R. Klein and A. Scholl, *Computing lower bounds by destructive improvement: An application to resource-constrained project scheduling*, European Journal of Operational Research **112** (1999), 322–346.

[64] R. Kolisch, *Integrated scheduling, assembly area and part-assignment for large-scale, make-to-order assemblies*, International Journal of Production Economics **64** (2000), 127–141.

[65] R. Kolisch and S. Hartmann, *Experimental investigation of heuristics for resource-constrained project scheduling: An update*, European Journal of Operational Research **174** (2006), 23–37.

[66] R. Kolisch, A. Sprecher, and A. Drexl, *PSPLIB-a project scheduling library*, European Journal of Operational Research **96** (1997), 205–216.

[67] O. Koné, C. Artigues, P. Lopez, and M. Mongeau, *Event-based milp models for resource-constrained project scheduling problems*, Computers & Operations Research **38** (2011), 3–13.

[68] A. Kooli, M. Haouari, L. Hidri, and E. Néron, *Improving classical energetic reasoning for the resource constrained project scheduling problem*, 12th international workshop on project management and scheduling (Tours, France), 2010.

[69] ______, *IP-based energetic reasoning for the resource constrained project scheduling problem*, Electronic Notes in Discrete Mathematics **36** (2010), 359–366.

[70] A. Kooli, M. Haouari, E. Néron, and J. Carlier, *Extension d'une borne inférieure préemptive pour le problème de gestion de projet à contraintes de ressources*, 13ème Congrès Annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (Angers, France), 2012.

[71] ______, *A preemptive bound for the rcpsp*, 13th international workshop on project management and scheduling (Leuven, Belgium), 2012.

[72] P. Laborie, *Algorithms for propagation resource constraints in ai planning and scheduling: Existing approaches and new results*, Artificial Intelligence **143** (2003), 151–188.

[73] A. Lahrichi, *Ordonnancements: La notion de parties obligatoires et son application aux problèmes cumulatifs*, RAIRO-RO **16** (1982), 241–262.

[74] J. Lee and Y. Kim, *A study of project scheduling optimization using tabu search algorithm*, Journal of the Operational Research Society **47** (1996), 678–689.

[75] O. Liess and P. Michelon, *A constraint programming approach for the resource-constrained project scheduling problem*, Annals of Operations Research **157** (2008), 25–36.

[76] V. Maniezzo and A. Mingozzi, *A heuristic for the multi mode project scheduling problem based on benders decomposition*, Handbook on Recent Advances in Project Scheduling, pp. 179–196, Kluwer, 1998.

[77] P. Martin and D. Shmoys, *A new approach to computing optimal schedules for the job-shop scheduling problem*, Proceedings of the 5th international Integer Programming and Combinatorial Optimization conference, Lecture Notes in Computer Science, 1996.

[78] A. Mingozzi, V. Maniezzo, S. Ricciardelli, and L. Bianco, *An exact algorithm for project scheduling with resource constraints based on a new mathematical formulation*, Management Science **44** (1998), 714–729.

[79] R. Möhring, A. Schulz, F. Stork, and M. Uetz, *Resource constrained project scheduling: Computing lower bounds by solving minimum cut problems*, Lecture Notes in Computer Science **1643** (1999), 139–150.

[80] ______, *Solving project scheduling problems by minimum cut computations*, Management Science **49** (2003), 330–350.

[81] I. Nabeshima, *Algorithms and reliable heuristics programs for multiproject scheduling with resource constraints and related parallel scheduling*, University of Electro-Communications (1973).

[82] T. Nazareth, S. Verma, S. Bhattacharya, and A. Bagchi, *The multiple resource constrained project scheduling problem: A breadth-first approach*, Journal of Operational Research **112** (1999), 347–366.

[83] K. Neumann, C. Schwindt, and J. Zimmermann, *Recent results on resource constrained project scheduling with time windows: Models, solution methods, and applications*, Central European Journal of Operations Research **10** (2002), 113–148.

[84] K. Neumann and J. Zimmermann, *Procedures for resource leveling and net present value problems in project scheduling with general temporal and resource constraints*, European Journal of Operational Research **127** (2000), 425–443.

[85] ______, *Exact and truncated branch-and-bound procedures for resource-constrained project scheduling with discounted cash flows and general temporal constraints*, Central European Journal of Operations Research **10** (2002), 357–380.

[86] K. Nonobe and T. Ibaraki, *Formulation and tabu search algorithm for the resource constrained project scheduling problem*, Essays and Surveys in Metaheuristics, pp. 557–588, Kluwer, 2002.

[87] E. Néron, *Du flow-shop hybride au problème cumulatif*, Ph.D. thesis, Université de Technologie de Compiègne, 2005.

[88] N. Nudtasomboon and S. Randhawa, *Resource-constrained project scheduling with renewable and non-renewable resources and time-resource tradeoffs*, Computers and Industrial Engineering **32** (1997), 227–242.

[89] J. Patterson, *A comparison of exact approaches for solving the multiple constrained resource project scheduling problem*, Management Science **30** (1984), 854–867.

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

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

[92] M. Ranjbar, B. de Reyck, and F. Kianfar, *A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling*, European Journal of Operational Research **193** (2009), 35–48.

[93] B. De Reyck and W. Herroelen, *A branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations*, European Journal of Operational Research **111** (1998), 152–174.

[94] E. Rolland, R. Patterson, K. Ward, and B. Dodin, *Scheduling differentially-skilled staff to multiple projects with severe deadlines*, European Journal of Operational Research (2005).

[95] W. Rom, O. Tukel, and J. Muscatello, *MRP in a job shop environment using a resource constrained project scheduling model*, Omega **30** (2002), 275–286.

[96] A. Schutt, T. Feydy, P. Stuckey, and M. Wallace, *Explaining the cumulative propagator*, Constraints **16** (2011), 250–282.

[97] A. Sprecher, *Resource-constrained project scheduling: Exact methods for the multi-mode case*, Lecture Notes in Economics and Mathematical Systems **409** (1994).

[98] ______, *Scheduling resource-constrained projects competitively at modest memory requirements*, Management Science **46** (2000), 710–723.

[99] A. Sprecher and A. Drexl, *Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm*, European Journal of Operational Research **107** (1998), 431–450.

[100] A. Sprecher, R. Kolisch, and A. Drexl, *Semi-active, active and non-delay schedules for the resource-constrained project scheduling problem*, European Journal of Operational Research **80** (1997), 94–102.

[101] P. Östergård, *A new algorithm for the maximum-weight clique problem*, Nordic Journal of Computing **8** (2001), 424–436.

[102] J. Stinson, E. Davis, and B. Khumawala, *Multiple resource constrained scheduling using branch and bound*, AIEE Transactions **10** (1978), 252–259.

[103] F. Tercinet and E. Néron, *Energetic reasoning and bin-packing problem for bounding a parallel machine problem*, 4'OR (2006), no. 4, 297–317.

[104] L. Tseng and S. Chen, *A hybrid metaheuristic for the resource-constrained project scheduling problem*, European Journal of Operational Research **175** (2006), 707–721.

[105] V. Valls, F. Ballest, and S. Quintanilla, *Justification and rcpsp: A technique that pays*, European Journal of Operational Research **165** (2005), 375–386.

[106] ———, *A hybrid genetic algorithm for the resource-constrained project scheduling problem*, European Journal of Operational Research **185** (2008), 495–508.

[107] J. van den Akker, G. Diepen, and J. Hoogeveen, *A column generation based destructive lower bound for resource constrained project scheduling problems*, CPAIOR, 2007, pp. 376–390.

[108] A. Viana and J. de Sousa, *Using metaheuristics in multiobjective resource constrained project scheduling*, European Journal of Operational Research **120** (2000), 359–374.

[109] J. Weglarz, J. Blazewicz, W. Cellary, and R. Slowinski, *An automatic revised simplex method for constrained resource network scheduling*, ACM Transactions on Mathematical Software **3** (1977), 295–300.

[110] J. Weglarz, J. Józefowska, M. Mika, and G. Waligóra, *Project scheduling with finite or infinite number of activity processing modes - a survey*, European Journal of Operational Research **208** (2011), 177–205.

[111] S. Wu, H. Wan, S. Shukla, and B. Li, *Chaos-based improved immune algorithm (cbiia) for resource-constrained project scheduling problems*, Expert Systems with Applications **38** (2011), 3387–3395.

[112] D. Yamashita, V. Armentano, and M. Laguna, *Robust optimization models for project scheduling with resource availability cost*, Journal of Scheduling **10** (2007), 67–76.

Résumé :

Le problème de gestion de projet à contraintes de ressources est un des problèmes les plus étudiés dans la littérature. Il consiste à planifier des activités soumises à des relations de précédence, et nécessitant des ressources renouvelables. L'objectif est de minimiser la durée du projet, soit le *makespan*. Nous étudions le problème de gestion de projet à contraintes de ressources. Nous nous sommes intéressées à la résolution exacte du problème. Dans la première partie de la thèse, nous élaborons une série de bornes inférieures basées sur le raisonnement énergétique et des formulations mathématiques. Les résultats montrent que les bornes proposées surpassent ceux de la littérature. Dans la deuxième partie, nous proposons des procédures par séparation et évaluation utilisant les bornes inférieures dévelopées dans la première partie.

Mots clés :

Gestion de projets à contraintes de ressources, Raisonnement énergétique, Formulation mathématique, Borne inférieure, Procédure par séparation et évaluation, Règle de dominance.

Abstract :

Resource Constrained Project Scheduling Problem is one of the most studied scheduling problems in the literature. It consists in scheduling activities, submitted to precedence relationship, and requiring renewable resources to be processed. The objective is to minimize the project duration, i.e., the *makespan*. We study the Resource Constrained Project Scheduling Problem. We are interested on the exact resolution of the problem. In the first part of the thesis, we develop a series of lower bounds based on energetic reasoning and mathematical formulations. The computational results show that the proposed lower bounds outperform the ones of the literature. In the second part, we propose Branch-and-Bound procedures using the lower bounds developed on the first part.

Keywords :

Resource Constrained Project Scheduling Problem, Energetic Reasoning, Mathematical formulation, Lower bound, Branch-and-Bound, Dominance rule.