Electric Vehicles Charging Scheduling Optimisation

Nhan Quy Nguyen

Thèse
de doctorat
de l'UTT

Nhan Quy NGUYEN

Electric Vehicle Charging Scheduling Optimisation

Spécialité :
Optimisation et Sûreté des Systèmes

2017TROY0024

Année 2017

utt
UNIVERSITÉ DE TECHNOLOGIE
TROYES

THESE

pour l'obtention du grade de

DOCTEUR de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

Spécialité : OPTIMISATION ET SURETE DES SYSTEMES

présentée et soutenue par

Nhan Quy NGUYEN

le 28 septembre 2017

Electric Vehicle Charging Scheduling Optimisation

JURY

M. C. ARTIGUES	DIRECTEUR DE RECHERCHE CNRS	Président (Rapporteur)
M. L. AMODEO	PROFESSEUR DES UNIVERSITES	Directeur de thèse
M. H. CHEHADE	DOCTEUR, CHERCHEUR ASSOCIE UTT	Examinateur
M. S. DAUZÈRE-PÉRÈS	PROFESSEUR ENSM SAINT-ETIENNE	Rapporteur
M. E.-G. TALBI	PROFESSEUR DES UNIVERSITES	Examinateur
M. T. USLÄNDER	DOKTOR-INGENIEUR	Examinateur
M. F. YALAOUI	PROFESSEUR DES UNIVERSITES	Directeur de thèse

Personnalité invitée

M. P. TOGGENBURGER	INGENIEUR

Acknowledgement

I am most grateful to the members of my committee, Prof. Stéphane DAUZERE-PERES and Prof. Christian ARTIGUES for their high-quality reports and though. I would also like to thank Dr. Thomas USLÄNDER, Prof. El-Ghazali TALBI for accepting to be my thesis referees. The committee's guidance, discussion, and feedback have been precious.

I want to express my sincere gratitude to my supervisors Prof. Farouk YALAOUI and Prof. Lionel AMODEO. You are my example of outstanding researchers, of experienced mentors and above all, you are my role-models.

I would like to thank Mr. Pascal TOGGENBURGER for offering me such an opportunity to work on this project, and for assuring me an excellent working condition.

Exceptional thanks go to my advisor, Dr. Hicham CHEHADE. Thank you for all your patience, your guidance and your support over the three years.

This work cannot be done without the support of all my friends and colleagues. Very special thanks go to Etienne POLLET, Samy JACQUET, Tan NGUYEN and Xixi WANG for being the best friends, co-workers and fellow PhD student one can hope for.

Finally, I lovingly dedicate this thesis to my Mother, Nhan Nghia NGUYEN, who have taught the very first lessons of life. Your constant love and encouragement have sustained me for always.

Contents

1 General introduction **19**

1.1 Study motivation and industrial context 19

1.2 Contributions 21

1.3 Dissertation contents 22

2 State-of-the-art **25**

2.1 Introduction 26

2.2 Electric vehicles charging coordination 26

2.2.1 Electric vehicles charging terminology 26

2.2.2 Literature reviews and classification 27

2.3 Scheduling problem with single additional resource 28

2.3.1 Terminologies and definitions 28

2.3.2 Optimisation criteria 35

2.3.3 Scheduling problem with additional resource notation 36

2.3.4 Mixed Integer Liner Programming 37

2.4 Literature review 39

2.4.1 The Discrete-Continuous Scheduling Problem 40

2.4.2 Parallel task scheduling problem (PTSP) 41

2.4.3 Cumulative Scheduling Problems (CSP) 42

2.5 Research positioning 43

2.6 Conclusion 43

3 EVCC problem classification and mathematical formulation **47**

3.1 Introduction 48

3.2 EVCC problem classification 48

3.2.1 Electric vehicle charging coordination (EVCC) constraints 49
3.2.2 ACPF-ACPV configurations 51
3.3 Complexity analysis 54
3.3.1 Notation and complexity 55
3.4 MILP formulation 56
3.4.1 Cumulative formulation 59
3.4.2 Disjunctive formulation 62
3.4.3 LP-Relaxation analysis 66
3.5 Computational study 68
3.5.1 Random instance generator 68
3.5.2 Numerical experiments 70
3.6 Conclusion 75

4 Exact solution approach: A Branch-and-Price algorithm 77

4.1 Introduction 78
4.2 Preliminaries 78
4.2.1 Column generation 78
4.2.2 Branch-and-Price 79
4.3 The branch and price algorithm 81
4.3.1 Decomposition methods 81
4.3.2 Mixed Integer Linear Programming (MILP) formulation 81
4.3.3 Local sub-problem formulation 82
4.3.4 Convexification approach 84
4.3.5 Linear relaxation of the master problem 85
4.3.6 Restricted master problem 87
4.3.7 The pricing problem 87
4.3.8 The branching scheme 88
4.4 Computational implementation 91
4.4.1 Pricing problem dual bound for earlier termination 91
4.4.2 Branching restriction enforcing on master and pricing problem 91
4.4.3 Initial solutions 93

4.5 Numerical tests . . . 94
4.5.1 Test instances generation . . . 94
4.5.2 Numerical results . . . 94
4.6 Conclusion . . . 97

5 Approximate solution approach: Heuristics HLA 99

5.1 Introduction and motivation . . . 100
5.2 Global procedure . . . 101
5.3 Initial solution construction phase . . . 102
5.3.1 Initial resource consumption: A reversed power bandwidth . . . 103
5.3.2 Greedy Best Fit . . . 104
5.3.3 Guillotine Cut and Layering . . . 108
5.3.4 Illustrative example . . . 109
5.4 Solution improvement phase . . . 113
5.4.1 Greedy occupation heuristic . . . 115
5.4.2 Shuffle heuristics . . . 117
5.4.3 Illustrative example . . . 119
5.5 Computational study . . . 120
5.6 Conclusion . . . 127

6 Simulation based forecast and online scheduling 129

6.1 Introduction . . . 130
6.2 Preliminaries . . . 131
6.2.1 Terminologies . . . 131
6.2.2 Rescheduling methodology . . . 132
6.3 Method outline . . . 133
6.3.1 Predictive-reactive rescheduling . . . 133
6.3.2 The power and cost trade-off . . . 134
6.4 The simulation based forecast . . . 136
6.4.1 Fixed time-horizon simulator . . . 137
6.4.2 Fixed power bandwidth simulator . . . 138
6.4.3 Case study . . . 139

6.5 Online scheduling . . . 143
6.5.1 From deterministic scheduling to online scheduling . . . 143
6.5.2 The decision making policies . . . 143
6.5.3 Reactive rescheduling . . . 144
6.6 Conclusion . . . 146

7 Software implementation **149**
7.1 Introduction . . . 149
7.2 The EV charging management description . . . 149
7.3 Software design . . . 150
7.4 Conclusion . . . 154

8 Conclusion et Perspectives **157**

A Résumé étendu en français **161**
A.1 Introduction generales . . . 162
A.1.1 Motivation et contexte industriel . . . 162
A.1.2 Contributions . . . 164
A.1.3 Contenu de la thèse . . . 165
A.2 L'état de l'art . . . 167
A.2.1 Introduction . . . 167
A.2.2 CRVE . . . 167
A.2.3 Problème de planification avec une ressource supplémentaire unique . 168
A.2.4 La revue de littérature . . . 168
A.3 Classification des problèmes CRVE et Formulation mathématique . . . 171
A.3.1 Classification des problèmes CRVE . . . 171
A.3.2 Formulation de PLNEM . . . 171
A.3.3 Formulation cumulative (conjonctive) . . . 174
A.3.4 Formulation disjonctive . . . 176
A.3.5 Analyse convexe . . . 179
A.3.6 Test numériques . . . 179
A.4 Résolution exacte : Algorithme Branch-and-Price . . . 181
A.4.1 Contexte . . . 181

A.4.2 Problème maître . . . 181
A.4.3 Problème de pricing . . . 183
A.4.4 Stratégies de branchement . . . 184
A.4.5 Bornes duales pour stabiliser la génération de colonne . . . 185
A.5 Interprétation des résultats des tests numériques . . . 186
A.6 Résolution approximative : Heuristique HLA . . . 187
A.6.1 Procédure globale . . . 187
A.6.2 Phase de construction de la solution initiale : Superposition de la ressource . . . 188
A.6.3 Phase d'amélioration de la solution . . . 188
A.6.4 Interprétation des résultats des tests numériques . . . 189
A.7 Prévisions basées sur la simulation et ordonnancement en temps réel . . . 191
A.7.1 Modèle prévisionnel . . . 191
A.7.2 Ordonnancement en temps réel . . . 194
A.8 Le logiciel . . . 196
A.9 Conclusion et Perspectives . . . 198

List of Figures

1-1 An Electric Vehicle (EV) charging parking illustration. *Source: ParknPlug* . . 20

1-2 An EV in charge. *Source: ParknPlug* 22

2-1 The simplified EV charging infrastructure 27

2-2 Comparison of three types of parallel jobs 35

3-1 A classification of EVCC constraints 49

3-2 Illustrations of 5 EVCC configurations with corresponding possible solutions . 54

3-3 A resource allocation example of job i 58

3-4 An example of the state of job 58

3-5 The illustration of the Charging algorithm with variable power (ACPV) 2 problem 60

3-6 Decision variable $y_{i,k}$ and $c_{i,k}$ 62

3-7 Decision variables $\alpha_{i,k}$ and $\beta_{i,k}$ with the processing of job i 64

3-8 The dashed zone is the feasible region of variables y and c in the LP-Relaxation of Conjunctive MILP formulation $(P - Cml)$ 66

3-9 The weighted line is the feasible region of variables α and β in the LP-Relaxation of Disjunction MILP formulation $(P - Dsj)$ 66

3-10 The feasible space of the LP-Relaxation of $P - Cml$ taken from constraints (3.14) and (3.15) 67

3-11 The feasible space of the LP-Relaxation of $P - Dsj$ taken from constraints (3.26) and (3.29) 68

3-12 The feasible space of the LP-Relaxation of $P - Dsj$ reinforced by cut (3.34) . 68

3-13 Test instance generation 69

3-14 *BTGap* found by 5 formulations throughout different problem sizes n. 73

4-1 The column generation algorithm 80

4-2 A branching example 89
4-3 The generic branching strategy for the ACPV 2 problem 92

5-1 Heuristic HLA 102
5-2 The initial resource consumption explication 105
5-3 Shiftable interval $J_i = \langle r_i, d_i, p_i \rangle$ 105
5-4 Maximum resource heights created by three placements of job i 106
5-5 Resource height accumulation of second placement ϕ_2 107
5-6 Resource height accumulation of third placement ϕ_3 107
5-7 Resource allocation of job 1 being planned by GBF algorithm. 111
5-8 Guillotine cuts of job 1: Initial iteration 111
5-9 Job 1's after-cut resource reallocation: First iteration 112
5-10 Job 1's after-cut resource reallocation: Second iteration 112
5-11 Resource allocation of job 2 found by GBF and guillotine cut $gcut_2$ 113
5-12 Job 2's after-cut resource reallocation: First and also last iteration 113
5-13 Job 3's resource allocation found by GB and guillotine cuts 114
5-14 Job 3's resource allocation: first shuffle and cut_1 found 114
5-15 Job 3's resource allocation: last cut 115
5-16 Initial solution of *Guillotine Cut and Layering* 115
5-17 Solution after *Rectangular-form shuffle* procedure 120
5-18 Solution after *Greedy Occupation* procedure 121
5-19 Solution after *free-form shuffle* procedure 121
5-20 *Gap* found by HLA throughout different problem sizes nH^2. The horizontal axes are in logarithmic scale. 123
5-21 Average $CPUTime$ throughout different problem sizes nH^2 126

6-1 The rescheduling framework classification 133
6-2 Overall description of Predictive-reactive rescheduling 134
6-3 A big power subscribed let the scheduling be completely on the off-peak hours 135
6-4 An earlier schedule start (left-shift) for small power subscribed 135
6-5 The simulation based forecast 136
6-6 Fixed time-horizon simulator description 137

6-7 Fixed power bandwidth simulator description . . . 139

6-8 Histogram of customer behaviour to justify the choice of random distributions on arrival times and charging demands. . . . 140

6-9 Histogram of the maximum total charging power of 30 EV resulted from 10000 tests . . . 140

6-10 The corresponding probability cumulative function F_{BW} and its distribution fit counterpart . . . 140

6-11 Expected shift (h) to start before off-peak hours to assure feasibility according to power bandwidth used ($n_{EV} = 30$ and $n_{tests} = 10000$) . . . 142

6-12 The online schedule policy . . . 145

6-13 The online schedule policy in example . . . 146

7-1 The schematic of the charging solution. *Source: ParknPlug* . . . 151

7-2 The information flow . . . 152

7-3 The CMU description . . . 154

A-1 Une illustration de stationnement des VE en charge. *Source : ParknPlug* . . . 163

A-2 Un VE en charge. *Source : ParknPlug* . . . 164

A-3 L'infrastructure de chargement simplifiée . . . 167

A-4 Variables de décision $y_{i,k}$ et $c_{i,k}$ . . . 176

A-5 Variables de décision $\alpha_{i,k}$ et $\beta_{i,k}$ . . . 177

A-6 Générateur aléatoire . . . 179

A-7 Un exemple de branchement . . . 184

A-8 Heuristique HLA . . . 187

A-9 Prévisions basées sur la simulation . . . 191

A-10 Description du simulateur d'horizon temporel fixe . . . 192

A-11 L'histogramme de la puissance de charge totale maximale de 30 VE résulte de 10000 tests . . . 192

A-12 La fonction cumulative de probabilité correspondante F_{BW} et sa répartition correspond à l'équivalent . . . 192

A-13 Description du simulateur de bande passante fixe . . . 193

A-14 Décalage prévu (h) pour commencer avant les heures creuses pour assurer la faisabilité en fonction de la bande passante utilisée ($n_{VE} = 30$ et $n_{tests} = 10000$) 193

A-15 La politique d'ordonnancement en temps réel . . . 194

A-16 La politique de l'ordonnancement en temps réel dans l'exemple . . . 195

A-17 Les flux d'informations 197
A-18 Le CMU 197

List of Tables

2.1 Classification of some EVCC approaches in the literature 29

2.2 Classification of some featured approaches on the theoretical scheduling problem under resource constraint 44

3.1 The Charging algorithm with fixed power (ACPF) - ACPV configurations . . 53

3.2 Constraints (3.14) and (3.15) explication 61

3.3 Random distributions parameters of generated elements for test instances . . 70

3.4 Solving times and best solving times gap resulted from different formulations 72

3.5 Number of nodes and MILP-Gap resulted from different formulations 74

4.1 Numerical tests results of **LFlex branching strategy** 95

4.2 Numerical tests results of **MFlex branching strategy** 96

5.1 Properties of jobs used in example 104

5.2 Total resource amount available in example 104

5.3 Summary of three LNS methods 119

5.4 Objective values of solutions found by heuristics: HLA-RAG, HLA-ARG, EDD, SPT and by solver CPLEX. 124

5.5 *CPUTime* of heuristics: HLA-RAG, HLA-ARG, EDD, SPT and of solver CPLEX. 125

6.1 Customer random behavior parameters: input for simulators 141

6.2 The decision making policies in resume 144

7.1 Event definitions 155

7.2 Events notations 155

A.1 Classification de certaines approches CRVE dans la littérature 168

A.2 Classification de certaines approches en vedette sur le problème de planification théorique sous contrainte de ressource 169

A.3 The ACPF - ACPV configurations 172

Abreviations

$P - Cml$ Conjunctive MILP formulation.

$P - Dsj$ Disjunction MILP formulation.

ACPF Charging algorithm with fixed power.

ACPV Charging algorithm with variable power.

EV Electric Vehicle.

EVCC Electric vehicle charging coordination.

EVSE Electric vehicle supply equipment.

GBF Greedy Best Fit algorithm.

HLA Heuristic of Layering and Adapting.

MILP Mixed Integer Linear Programming.

MinPeak Power peak minimisation.

SOS1 Special Ordered Sets of type 1.

Abbréviations en Français

$P - Cml$ FormulationPLMNE Conjonctive.

$P - Dsj$ Formulation PLMNE Disjonctive.

ACPF Algorithme de charge avec puissance fixe.

ACPV Algorithme de charge avec puissance variable.

CRVE Coordination de recharge des véhicules électriques.

GBF Algorithme Greedy Best Fit.

HLA Heuristique de la mise en couches et de l'adaptation.

PDC Point de charge.

PLMNE Programmation linéaire mixte en nombre entier.

SOS1 Ensembles commandés spéciaux de type 1.

VE Véhicule électrique.

Chapter 1

General introduction

Contents

1.1 Study motivation and industrial context 19

1.2 Contributions 21

1.3 Dissertation contents 22

1.1 Study motivation and industrial context

This thesis was carried out in the context of an Industrial Convention of Research Formation (CIFRE) with the cooperation between the **Laboratory of Industrial Systems Optimisation** (LOSI) and the company **Park'nPlug** (PnP). Hence it is important to underline the industrial context, which motivates and orientates the centre of studies during this thesis.

The problem that arises in this thesis is the optimisation of the charging scheduling of electric vehicles (EV), which is also known as Electric vehicle charging coordination (EVCC) problem. In the last ten years, we have observed the big jump in the electric vehicle technologies. The battery density increases by 400% in 7 years (2008-2015) while the cost per kWh is reduced by 73% [17]. As a consequence, the number of electric cars which had been less than a thousand in 2005, attends 1.26 million in 2015 [17]. Electric Vehicles Initiative (EVI) sets an objective of 20 million EV deployed by 2020. In the middle of the ecological

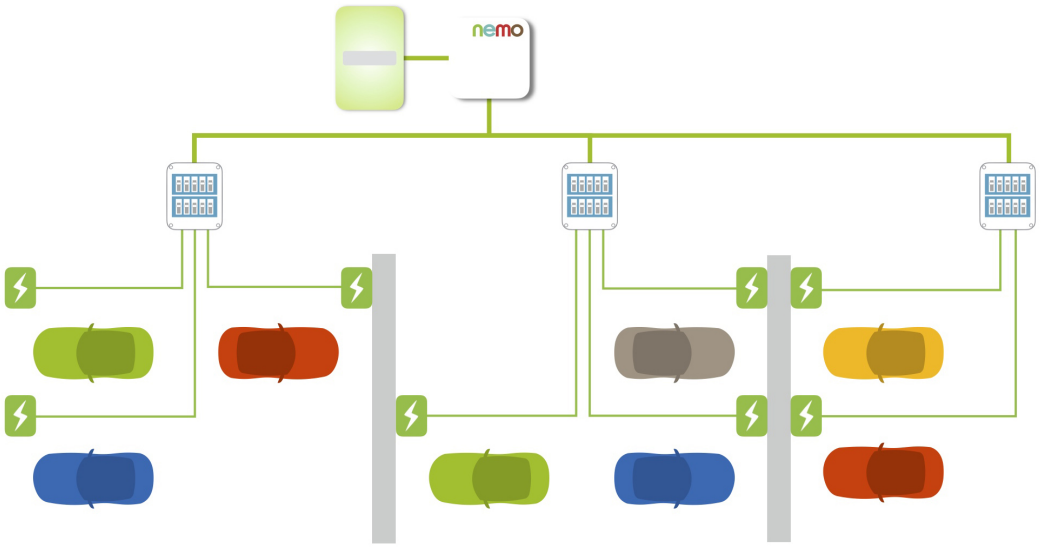


Figure 1-1 – An EV charging parking illustration. *Source: ParknPlug*

and energy crisis, a shift in transportation from fossil consumed vehicles to electric vehicles becomes a sustainable solution. This change can reduce the carbon emission and assure the energy security. In addition, stand behind the success of EV deployment is the advantageous governmental policies and the awareness of EV user on the sustainable development. However, EV charging is an enormous power consumption task over an extended period. For single phase charge point, charging rate can vary from 3.3 to 7.4kW for a charging duration from 3 to 6 hours. A three-phase charger can provide from 10 to 22kW for 1 to 3 charging hours [19]. Especially, there is also ultra-fast charge rate with a power of 43kW. Hence, the high penetration of electric vehicles can cause many concerns to the electric grid. Also, the power subscription for individual charge point would be wasteful, since the charging time takes in average 17% of the availability time. In additional, a personal charge point and its electrical cost could be costly for EV user. As a good solution to those issues, a collective and coordinated charging process can stabilise the grid and optimise the charge point productivity, also minimise the electrical cost.

The **Park'nPlug** company is located in France and provides the corporate EV recharging services and management. In France, the private sector (residential, office parking) dominates with a share of 85% of the charge points compared to 15% share of the public sector (supermarket, public, highway parking)[17]. The main difference between those two areas is the level of knowledge of the input data. In the private parking, we know all the

clients and their past behaviours whereas the information customer of the public sector is entirely unknown. Hence, the optimisation method on the charging scheduling would be completely different while facing each of the two problems. In the limit of this thesis, we tackle exclusively the private sector EV charging coordination. The electric cost is comparable with any other cost in the scheduling problem: it consists of a fix cost and a variable cost. The fixed cost is a monthly subscription and concerns the total bandwidth. Many French electrical providers propose two pricing plans: whether the energy cost is constant or it consists two sub-plan, the off-peak cost (cheaper) and the on-peak cost (more expensive). With the first option, the constant price, we have few possibilities to optimise the variable cost. Yet, the second plan requires an optimal usage schedule. One has to face the trade-off between the fixed cost and the variable cost. A narrow power subscription is cheap, however, takes a longer time to charges all the EV which can extend the charging time on the on-peak hours, thus increases the variable cost. So the minimisation of electrical cost should answer to two questions:

1. How much electrical power one should subscribe dedicated to the EV corporate charging to minimise the fixed cost?
2. When should we start the charging procedure to minimise the variable cost.

After answering to those questions, we move to the next phase: the optimisation of the service quality. One should liberate as soon as possible all the charging tasks in the waiting queue, with respect to all the constraints regarding the EV availability on the parking, the daily charging demand and the costumer's departure references.

1.2 Contributions

The first contribution of this thesis lays in the creation of a methodology to solve the industrial problem: A classification on the EVCC constraints and the definition of a novel class of the scheduling problem under single additional resource. This class of scheduling problem are expressed in 5 configurations namely ACPF 1/2 and ACPV 1a/1b/2.

The second contribution is the formulation of the EVCC problem. We propose two

Figure 1-2 – An EV in charge. *Source: ParknPlug*

mathematical formulations, followed by a detailed analysis on their LP-Relaxation.

The third principal contribution consists of two methods of resolution of the EVCC problem. We propose the exact resolution procedure by the Branch-and-Price Algorithm. We introduce with this approach the tuning up tools and branching strategies. Also, we design a family of constructive heuristic named Layering and Adapting (HLA), based on the Large Neighbour Search and Strip packing principles, which can solve large instances of the problem efficiently in a very short execution time.

The fourth contribution is the conception of a simulation based forecast and an online scheduler to cope with schedule's uncertainties and real-time events. It is suitable for the technical implementation thanks to its reconfigurability and implementability.

Finally, we deploy all the algorithmic research done by creating a stand-alone scheduler software. This software is autonomous and is implemented in a microprocessor which provides the smart charging function for the charge point.

1.3 Dissertation contents

In this first chapter, we point out the motivation and the context of the thesis. To provide a theoretical background for our scientific development made in this thesis, we

outline in chapter 2 the terminologies and definitions of the scheduling problems under single additional resource, the electric vehicle charging coordination problem and the MILP formulation technique. We review the existed works, identify the key research gap so we can position our research.

To provide the methodology to tackle the problem, in chapter 3, we identify the electric vehicle coordination constraints, according to five sets of constraints. Then we introduce five configurations of the scheduling problems. In the remaining of the chapter, we formulate the most generic configuration of the scheduling class, the ACPV 2 in two MILP formulations: a conjunctive formulation and a disjunctive formulation. To have a more profound knowledge of the problem, we make a convex hull analysis of the two formulations and introduce two valid cuts. We test at the end of the chapter the exact resolution of those two developed models.

Both chapter 4 and chapter 5 are dedicated to the solution procedure of the problem. In chapter 4, we state that the disjunctive formulation of the problem consists of a block matrix. For that reason, we can apply the Danzig-Wolfe decomposition to partition the formulation. For that reason, we introduce the Branch-and-Price Algorithm as an exact approach to solving the EVCC problem. In search of a faster algorithm to cope with larger size instances, by chapter 5, we design the dedicated Layering and Adapting heuristics to solve the problem. All the resolution methods are tested numerically so that we can investigate their computational strength.

To cope with schedule uncertainties, we first introduce in chapter 6 a simulation based forecast. The forecaster decides how much electrical power one must subscribe at the strategic level. Also, it makes precision on when does one has to start the daily charging procedure. Furthermore, we present the conception of an online scheduler. This conception brings deterministic algorithm to cope with the real-time event by the partial rescheduling policy.

In chapter 7 we make concrete the scientific research into the conception of a deployable stand-alone scheduling software. This software is implemented on the microprocessor at the EV charging management centre of the parking.

At the end of the thesis, we resume the results of our works, hence drawing conclusions. Finally, we can state the perspective and future works as the outlook of this thesis.

Chapter 2

State-of-the-art

Contents

2.1 Introduction 26

2.2 Electric vehicles charging coordination 26

2.2.1 Electric vehicles charging terminology 26

2.2.2 Literature reviews and classification 27

2.3 Scheduling problem with single additional resource 28

2.3.1 Terminologies and definitions 28

2.3.2 Optimisation criteria 35

2.3.3 Scheduling problem with additional resource notation 36

2.3.4 Mixed Integer Liner Programming 37

2.4 Literature review 39

2.4.1 The Discrete-Continuous Scheduling Problem 40

2.4.2 Parallel task scheduling problem (PTSP) 41

2.4.3 Cumulative Scheduling Problems (CSP) 42

2.5 Research positioning 43

2.6 Conclusion 43

2.1 Introduction

In this chapter, we present the basic terminologies and definitions for further developments in this thesis. First, we review the electric vehicle charging problem with its general definitions and conceptions. Then, we identify the corresponding field of study, which is the scheduling problem with a single additional resource, to tackle the industrial problematic. Concrete definitions of the machines, the tasks, and their characteristics help us to state all the explication and explanation made in this thesis. Also, we introduce the optimization criteria in terms of time and resource. An important part of this chapter is dedicated to the reviews and classifications on existed works which underline the major issues. Thereby, we can position our problem with existing works, create the definition of new classes of scheduling problems which can fill those major research gaps.

2.2 Electric vehicles charging coordination

2.2.1 Electric vehicles charging terminology

Through this section, we introduce some technical terms and definitions concerning the electric vehicle charging briefly. Regarding the charging infrastructure, an electric vehicle charging station supplies electricity for the charging of electric vehicles. The EV Charging Station is usually mentioned as an Electric vehicle supply equipment (EVSE). A Charge Point (CP) controls the power of one or several EVSE. The Central or Back-Office controls the behaviours of the charge points in the parking. The types and designs of the charging central are diverse and correspond to the service company who develops the charging solution. In contrary, the charge point designs are very standard. There are two major protocols to take into consideration: the charging standard which controls the energy loading between the EV charger and the EVSE and the communication protocol between the back-office and the charge point. The simplified charging infrastructure is introduced in figure 2-1.

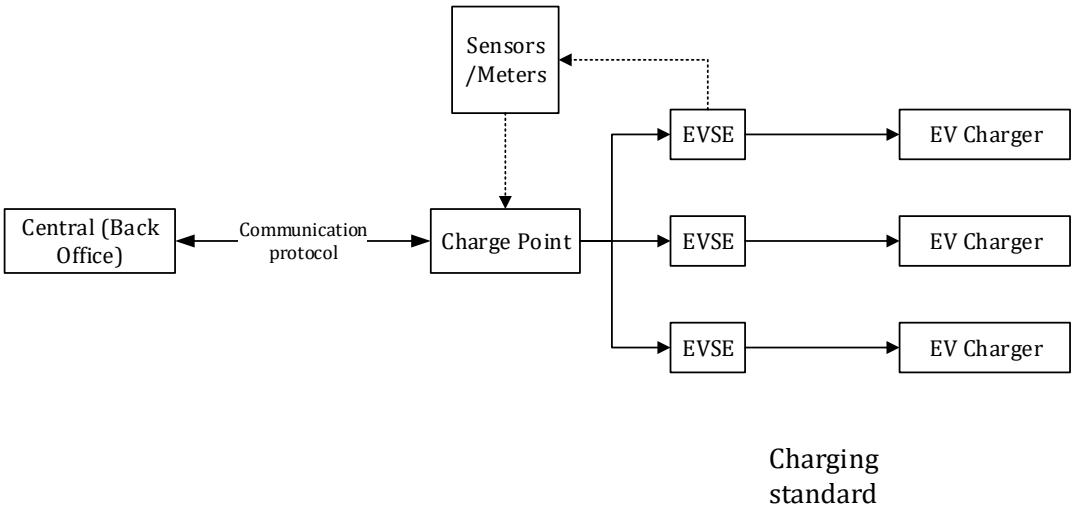


Figure 2-1 – The simplified EV charging infrastructure

2.2.2 Literature reviews and classification

There are two majors approaches to solve the EV coordination problems in the literature that we may define the local approach and the global approach. For the local approach, the solution procedure is implemented locally, usually at the charge point. For every new event, for instance, the end of a charge or a plugging of a newly arrived EV, the charge point will decide what to do next by itself. The decision is often driven by some based-rule, then it is called a "what-if" decision. The method can react quickly to real-time situations thanks to its simplicity. However, because the input problem is not treated globally, the overall scheduling performance is not sufficient. To cope with problems containing temporal constraints, this method could lead to an infeasible solution. Since the solution procedure is based on a little part of the information, then it can be called the implicit method. The majority of the implicit methods are based on the theory of control. Maasmann et al. [48] developed a feedback controlled fuzzy algorithm to balance charging power with respect to the user habits i.e. temporal constraints. Also, Faddel et al. [23] developed another feedback controlled fuzzy algorithm intended to maximise the total profit: the difference between the income from the selling of electricity to clients and the outcome from the buying of electricity from providers. Using another rule, Al-Awami et al. [4] introduce a charging voltage based controller to reduce the charging rate subject to the end-time of charge reference (i.e.

charging task expected deadline). Álvarez et al. [6] presented four variations of decentralised controller to serve the power balancing objective.

The second major solution procedure for the EVCC problem is the global scheduling method. This approach is called global because it takes the input as a whole. By using this approach, solutions found may have better performances compared to the solution obtained by the local method. The global scheduler can have a greater probability to find feasible solutions, even with temporal constraints and time-dependent resource, compared to the local approach. However, because it has to take the input altogether, this method finds difficulties to deal with uncertainties. Also, the computational time taken to decide is much longer than the time required for a rule-based decision. The global approach can be called sometime the explicit approach. In the limit of this chapter, we try to cite only some relevant works using the EVCC global approach. Concerning the meta-heuristic solution procedure, Alonso et al. [5] and Lee et al. [46] developed genetic algorithms to deal with the EVCC problem to minimise the total electric consumption and subscription cost. Pedrasa et al. [59] use the Particle Swarm Optimisation method to coordinate the EV charging scheduling with others household, optimising the total electric consumption cost. In the same context of the smart home energy management, Adika and Wang [2] introduced a demand side management algorithm, with the objective to flattening the electric power load. Karbasioun et al. [44] introduced a power strip-packing algorithm. This heuristic aims to minimise the power peak. Regarding the stochastic optimisation, Iversen et al. [35] proposed a Markov chain formulation to deal with randomness behaviours of EV client, then propose a stochastic optimisation schedule for the EVCC problem. We resume all the mentioned works in table 2.1.

2.3 Scheduling problem with single additional resource

2.3.1 Terminologies and definitions

The EVCC problem is a parallel machine scheduling problem. The charging processes are the tasks and the charge points are the machines. Machine scheduling problems decide the allocation of a set of tasks into a set of machines to optimise (maximisation or

Coordination method	**Resolution approach**	**Works**	**Objective**		
			Cost minimization	Power levelling / Load flattening	Time criteria minimization
Local (implicit) scheduling	Feedback controlled fuzzy algorithm	[48]		x	
		[23]	x		
	Voltage-based controller	[4]		x	
	Decentralized controller	[6]		x	
Global (explicit) scheduling	Genetic algorithm	[5, 46]		x	
	Particle swarm optimisation	[59]	x		
	Demand side management	[2]		x	
	Stochastic scheduling	[35]	x		
	Power strip packing	[44]		x	

Table 2.1 – Classification of some EVCC approaches in the literature

minimisation) a given objective and subject to a set of constraints. The solution procedure for the problem decides when a job starts and on which machine this job should be executed. There are many variations of the machine scheduling problem due to the different machine environments, scheduling criterion and additional constraints, as well as the divergence of the objective functions.

Machine environments

One can come up with four major types of machines: **single machine, identical machines, related machines and unrelated machines**. The single machine scheduling problem is made up of one machine; then the decision is reduced only to when a job can start. The machines can be identical, i.e. the processing speed of tasks is identical on each machine. In related machine environment, each machine has a fixed processing speed that is valid for all tasks. The last machine environment is the unrelated machines: the coupling of each pair task-machine creates a different speed. The complexity of unrelated machines problems dominates the related machine's problems, and the complexity of this latter dominates the identical machine's scheduling problem. The standard three-field notations [30] denotes the machine environment in the following way:

- 1: single machine
- P_m: identical machines
- Q_m: related machines
- R_m: unrelated machine

Jobs (Tasks, Activities)

Jobs contain different properties and constraints. The first three constraints are precedence constraints, stages constraints and temporal constraints. **Precedence constraints** restrict jobs to be executed in a precedence order, that indicates each task has a set of predecessors and a set of successors. The starting of a job requires the completion of all its

predecessors. This restriction is known commonly in project scheduling problems. Otherwise, jobs are called **independent**. With **stage constraints**, jobs have to be processed in multiple machines with respect to an order of stages. Scheduling problems with job stages are called **job-shop scheduling problems**. Without job stage, jobs are called **atomic**. **Temporal constraints** contain **release dates**, **deadlines** and **due-dates**. Any job cannot start before its release date and must end before its deadline (strict deadline). The due date is technically different from the deadline since it is acceptable for a job to finish after its due date. However, the difference between the completion time of a job and its due-date is called the **tardiness**. **Hard time-windows** constraints incorporate the release date and strict deadlines. **Soft time-windows constraints** consist of jobs' release dates and due dates. In this thesis, hard time windows constraints are denoted simply time-window constraints. Also, in the case of jobs having different importance, we denote its significance by a scalable: **weight-of-job**. Each task also has a measurable **processing time**. The standard notations of job properties are:

- C_i: the completion time of job i
- p_i: the processing time of job i
- r_i: the release date of job i.
- $C_i - r_i$: the flow time of job i
- $C_i - r_i - p_i$: the waiting time of job i
- d_i: the deadline (strict) for job i
- $\bar{d}_i$: the due date (no strict) of job i
- $\max\{C_i - \bar{d}_i, 0\}$: the tardiness of job i
- w_i: the weight of job i

There are two important properties of jobs that one should take into account: **the parallelism** and **the preemption**. If many machines can process a job at a time, the job is then called **parallel**. The **fixed parallelism** consist of jobs requiring a constant number of the machines during the entire processing. If the number of machines processing

a given job can vary over time, this job is then **malleable**. **Preemption** (activity splitting) allows the jobs to be interrupted at a machine and continued on another machine. On the contrary, once the non-preemptive job has started, it cannot be interrupted until it reaches the completion time.

Additional Resources

Besides the machines, which are also the discrete resources, the jobs may also need an additional resource to be processed. Additional resources can be classified according to their **divisibility** and **renewability**. Regarding the **divisibility**, additional resources can be discrete or they can be continuous. The discrete resources are machine resources, human resources... The continuous resource can be the electrical resources, heat flows... Furthermore, a resource is called renewable if it is available for each period with a constant (or variable) amount whereas a non-renewable resource is finite for the whole scheduling horizon. For instance, machine and human are discrete and renewable resources. The investment (money) can be considered as a continuous and non-renewable resource. In some particular cases, resources are both renewable and non-renewable which is called doubly-constrained resources. For instance, the energy stored in a laptop battery is renewable with a fixed maximum power and non-renewable since it has a total limit capacity.

Classically, the processing time of jobs is supposed to be constant, or, to be dependent only to the pairing of machine-task. However, the resource consumption may change the way the processing time of a job behaves. If the resource consumption is a constant, then jobs' processing time is also constant. On the contrary, if the resource consumption of a job is modifiable, then this latter plays an active role in the scheduling problem by adding additional decision variables to the problem.

Controllable processing times

The processing times are controllable when the distribution of resources to jobs can shorten or extend their processing times. In the basic model, the amount of the resources is fixed during the whole job processing. Consider a single resource scheduling problem, let

u_{ij} be the resource allocation to job i when it is paired with machine j (or u_i in the case of identical machines). The processing time p_{ij} is calculated by a resource consumption function $f_{ij}(u_{ij})$. In the majority of the studies of controllable processing time problems, the resource consumption function is a bounded linear function. It is also called compression function, which takes the form:

$$p_{ij} = f_{ij}(u_{ij}) = \bar{p}_{ij} - a_{ij}u_{ij} \quad 0 \leq \bar{u}_{ij} \leq \bar{p}_i j/a_{ij} \tag{2.1}$$

In this equation, $\bar{p}_{ij}$ is called the compressed processing time (initial processing time) for job i on machine j. $\bar{u}_{ij}$ is the upper bound of the amount resource can allocate to job i on machine j. a_{ij} is a positive linear coefficient denotes the compression rate of the resource. We can find another common resource consumption function which takes the form of a convex function:

$$p_{ij} = f_{ij}(u_{ij}) = \left(\frac{\tilde{x}_{ij}}{u_{ij}}\right)^k = \left(\frac{\tilde{x}_i}{h_{ij}u_{ij}}\right)^k \tag{2.2}$$

Where $\tilde{x}_{ij}$ is the workload of job i on machine j and k is a positive constant which is less than 1. We denote $\tilde{x}_i = \tilde{x}_{ij}/h_{ij}$ where h_{ij} is a positive constant, which stands for the resource consumption efficient of job i on machine j and $\tilde{x}_i$ is the workload of job i.

In the case of the identical machine scheduling problems, one can remove the index j from the parameters and variables. In the case of the multi-resource allocation problem, one may add the index l to the parameters and variables to denote the type of the used resources.

Controllable processing rates

In the controllable processing times problems, we can find that the resource consumption functions are time independent. Conversely, the time-dependent resource consumption function affects the speed of jobs instantaneously. In this kind of scheduling problems, the expected processing time of a job is time-varying. To formulate the processing rate, one

defines the processing state of jobs, or simply, the state of jobs. Consider the identical machine environment with the single additional resource, the state of job i at time t is denoted by $x_i(t)$, which is a non-decreasing function of time t (we suppose that accumulated state cannot deteriorate). The final state of job i, or the workload, is denoted by $\tilde{x}_i$. The completion time of a job i, which is unknown in advance, then $x_i(C_i) = \tilde{x}_i$. The processing rate of job i is the derivation of processing state of job i, which follows the resource consumption function $f_i(u_i(t))$ [80]

$$\dot{x}_i(t) = \frac{dx_i(t)}{dt} = f_i(u_i(t)) \quad x_i(0) = 0 \quad x_i(C_i) = \tilde{x}_i \tag{2.3}$$

In the time-discrete model, this differential equation becomes a difference equation:

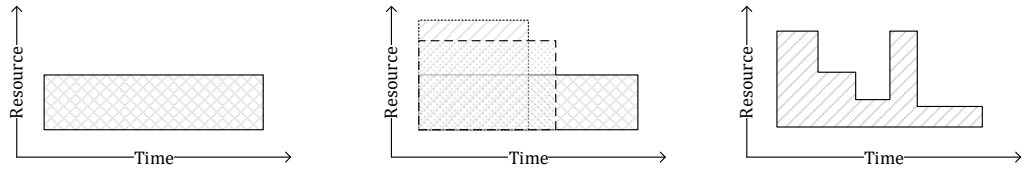
$$x_{i,k+1} - x_{i,k} = f_i(u_{i,k}) \quad x_{i,0} = 0 \quad x_{i,C_i} \geq \tilde{x}_i \tag{2.4}$$

The most studied resource consumption is a convex function: $f_i(u_i(t)) = h_i.u_i(t)^k$ with $h_i \in (0, 1]$ and $k \geq 1$. A special case when $k = 1$ we get the linear bounded resource consumption function. The special case with $h_i = 1$ and $k = 1$ defines the cumulative scheduling problem where the state of job is the accumulation of resource upto a given time.

Job scheduling problems

If the number of machines is greater than or equal to the number of jobs, the machine scheduling problem becomes the jobs scheduling problem (i.e. parallel scheduling problem). With a single additional resource, the way a job consumes a resource formulates three parallel scheduling problems:

Rigid jobs The resource is constantly required to process jobs, this requirement is a parameter of the input problem. Geometrically, jobs are rigid rectangles with given heights. The rigid job scheduling problem is the simplest job scheduling problem in term of complexity.



(a) Rigid job (b) Moldable job (c) Malleable job

Figure 2-2 – Comparison of three types of parallel jobs

Moldable jobs This type of job also requires a fixed amount of resource to process. However, unlike the rigid jobs, this amount of resource can be decided by the scheduler. The moldable jobs scheduling problem is notably close to the controllable processing time scheduling problem with a linear resource consumption function.

Malleable jobs Unlike the other two first types of jobs which have static resource allocation, malleable jobs can have dynamic resource allocation. Malleable jobs adapt the resource consumption at each decision interval according to the changing of the total available amount of resource. Jobs are flexible. Hence the decision of resources allocated to them should make at each interval.

2.3.2 Optimisation criteria

Criteria set F1: Time optimisation criteria With reference to the machine parallel scheduling problem, the two known optimisation criteria are total completion time minimisation and makespan minimisation. First, the total completion times minimisation denoted by $\sum(C_i)$ optimise the service quality. This criterion can also be expressed as mean flow-time minimisation $\frac{1}{n}\sum_{i=1}^{n}(C_i - r_i)$ to underline its user-benefit orientation. In the case of jobs (users) having different degrees of importance or benefits, one can minimise the total weighted completion time $\sum(w_i C_i)$. There are some variations of this criterion which are commonly researched such as total waiting time minimisation $\sum(C_i - p_i - r_i)$, total tardiness minimisation $\sum(\max\{C_i - \bar{d}_i, 0\})$. If the processing time is fixed and the release dates and the due dates are non-arbitrary parameters, those two minimisation criteria are equivalent to the total completion time minimisation since $\sum r_i - p_i$ and $\sum \bar{d}_i$ are constant. Second,

the makespan (C_{max}) minimisation, which aims to minimise the completion time of the last job in the system, optimise the production quality.

Criteria set F2: Resource optimisation criteria With reference to the scheduling problem with additional resources, one can first find the resource levelling problem in the majority of the concerning works. Resource levelling tends to minimise the variation of resource consumption over time. Resource levelling also implies to reduce the excessive resource demand (i.e. resource peak). However one may find in some situations that there is a trade-off between the resource peak and the total resource variation sum (the sum of the difference of resource usage between two consecutive periods). Then, there is a basic criterion which minimises only the resource peak, formally defined as the maximum resource consumption of all tasks at a given period. The second common objective is resource consumption cost minimisation. If the resource price is time-dependent, it is crucial also to decide right amount of resources at a given to minimise the overall consumption cost. Hence, we can derive four classes of optimisation problem:

- Optimise `F1` w.r.t. `F2`: time optimisation problem subject to an upper bound of resource.
- Optimise `F2` w.r.t. `F1`: resource optimisation problem subject to an upper bound of makespan.
- Optimise total cost of `F1` and `F2`: optimise the total cost of time and resource: $\alpha F_1 + \beta F_2$
- Multi-ojectives optimisation of `F1` and `F2`.

2.3.3 Scheduling problem with additional resource notation

Graham et al. [30] introduced a standard three field notation to classify scheduling problem: $\alpha/\beta/\gamma$. The α field indicates the machine environment. The β field describes the set of job properties and constraints. The γ field defines the optimisation criteria. To notate the properties of the resource, Blazewicz et al. [11] introduced an expansion to the β

field: $res\lambda\sigma\rho$. λ is a positive natural number stands for the number of resources. σ stands for the upper bound of the total amount of available resource, i.e. resource size. Finally, ρ represents the resource requirement upper limit of the tasks. If any of the three values for λ, σ and ρ are arbitrary or are considered as an input, we denote this parameter by a dot ($\bullet$) instead of using a positive number.

To close this subsection, we introduce three types of scheduling problems: deterministic scheduling problem, online scheduling problem and stochastic scheduling problem. In deterministic scheduling, all the parameters and input of the problem are known before the solution procedure (i.e. all known at time zero). In the online scheduling problem, the inputs of each job i are unknown until its release date (i.e. submission overtime scheduling). The online scheduling has two subclasses; the first one has the job workload or processing time known at submission. The second has the job workload or processing time unknown in advance (i.e. non-clairvoyant scheduling). Regarding the stochastic scheduling problem, we only know the random distributions of job parameters.

2.3.4 Mixed Integer Liner Programming

Formulation techniques

This part is dedicated to making a tour of the MILP state-of-the-art formulations techniques. The MILP has become one of the most common way to formulate the scheduling problem. There exist many ways to make a MILP model. The choice of a proper formulation is crucial since it can, indeed, influence the performance of the solver. Vielma [77] reasoned the success of MILP is due to the development of modern linear programming solver and the flexibility of the model itself. He also indicated that the well constructed MILP could boost up the solution procedure of the state-of-the-art solvers. Since most of the solvers are based on the branch-and-bound algorithm, the convex hull of each formulation is decisive to the performance of the solvers. Each formulation modifies the convex hull of the linear programming relaxation (LP-relaxation). The LP-Relaxation of a MILP formulation is a Linear Programming Formulation that replaces any integer constraints of the original MILP formulation to weaker constraints that each integer variable can be real (and can

be bounded). Technically, the tighter the hull, the better the formulation. Conventionally, one can say shortly that the formulation is tighter (or stricter) than another one. Also, a good MILP formulation can give a better idea of the solution procedure of the problem. Hooker [32] introduced a principled approach to MILP modelling. He stated that the MILP modelling consists of an identification of the disjunctive modelling and the knapsack constraints modelling. The author indicated that the disjunctive formulation is often stronger when one compare the performance of this formulation with other counterparts. Technically, disjunction means the union of many little polyhedra, each of those represents a solution possibility. Moreover, the knapsack model is used to express the counting ideas of integer variables which differs from the previous disjunction approach Hooker [32].

MILP Terminology and definition

To understand further presentation of the MILP in this thesis, we would provide some concerned fundamental definitions and terminologies:

MILP problem A mixed integer linear programming problem has the standard form:

$$\begin{aligned} \text{minimise } or \text{ maximise} \quad & cx + dy \\ \text{subject to} \quad & Ax + Gy \leq b \\ & x \in \mathbb{R}^n \text{ and } y \in \mathbb{Q}^p \\ \text{with} \quad & c \in \mathbb{R}^n; d \in \mathbb{R}^p; A \in \mathbb{R}^{m \times n}; G \in \mathbb{R}^{m \times p}; b \in \mathbb{R}^m \end{aligned} \tag{2.5}$$

LP-relaxation of a MILP The LP-relaxed problem takes the form of the original MILP, with variable y restricted by a weaker constraint where y is a positive real variable $P = (x, y) \in \mathbb{R}^n_+ \times \mathbb{R}^p_+ : Ax + Gy \leq b$.

Polyhedron Is the set $\{x|Ax \leq b\}$ for a matrix $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $x \in \mathbb{R}^n$. A bounded polyhedron is called polytope.

Feasible region of MILP The set S of feasible region of the MILP is defined by $S := P \cap \mathbb{Z}_+^n \times \mathbb{R}_+^p$

The convex hull of MILP the convex hull of the MILP is noted $conv(S)$ which is the smallest convex set containing S.

Special Ordered Sets of type 1 (SOS1) are a set of binary variables, at most one of which can take the one value, the others are strictly being zero. If $x = \{x_1, \ldots, x_n\}$ is a SOS1 then $\sum_{i=1}^{n} x_i = 1$ and $x_i \in \{0, 1\} \quad \forall i$

2.4 Literature review

We first position our problematic in the context of theoretical scheduling problems. The EV charging is a resource-consumption task. The operator (machine) is the EVSE. Hence the machine environment is considered to be identical. Charging tasks require a single additional resource which is electrical power. The amount of energy accumulated at a given time decides the processing time. If the charging power is time-dependent, then the processing rate is controllable, or, the charging tasks are malleable. Otherwise, the controllable processing time should be considered, corresponding to the moldable task scheduling problem. Because of that position of our industrial problematic, we review followingly works concerning the controllable processing time/rate scheduling problem and the parallel tasks scheduling problems, precisely the moldable and malleable tasks scheduling problems. The controllable processing time raised in the practical issue of the allocation of a shared common resource. Let us take a classical example of this type of scheduling problems, Janiak [36] investigated the ingot preheating process in steel mill problem. The common shared resource is the gas flow intensity. The more intense the gas flows to a given ingots, the hotter it would be. The increasing on the temperature of the preheated ingot can reduce its processing time in steel mills. The problems were actively studied in the 90s, mainly dedicated to solving heavy industrial problematic and the project scheduling problem with an additional resource [75]. We can find the continuous mathematical formulation and an exact solution procedure. Nowicki and Zdrzałka [58] formulate the trade-off between pro-

cessing time and resource consumption with compression function (2.1) by the compression cost integrated into the minimisation. Most of the controllable processing time scheduling problem, especially problems with temporal constraints, are NP-Hard [66]. However, when the environment is single machine or tasks are preemptive, the complexity could be solved in a polynomial time.

2.4.1 The Discrete-Continuous Scheduling Problem

Relating to the controllable processing rate, a specific class of the problem is defined and called Discrete-Continuous scheduling problem (DCSP). The DCSP problem is formally defined by [80], consisting of the programming of n tasks to m identical machines with a divisible and renewable resource. The term discrete comes from the machine's environment, and the term continuous originates from the property of the additional resource. According to our knowledge, this is the most generic and structured formulation so far to model the processing speed by resource consumption function. Nonetheless, the DCSP gained less attention in nearly two decades, until the beginning of the 2000s, when the energy reasoning, power-aware scheduling and the parallel computing both became serious issues. Since then, many works have been done regarding the convex and no greater than linear resource consumption function of job i takes the form $f_i = h_i u_i^{1/a}$ with a is positive constant and h_i is a linear coefficient, often called resource consumption efficiency. The DSCP problem with temporal constraint and non-preemptive tasks is NP-Hard [41]. Gorczyca and Janiak [28] studied the specific DCSP problem named PU_{max} which minimise the resource peak on an upper bound of makespan (problem optimising `F2` w.r.t. `F1`). They proposed five optimal properties; among them, one valuable property on the optimal allocation of resource given the jobs sequence. In agreement with this properties, they establish an exact resolution and an approximative approach to solving the problem.

Regarding the second problem to optimise time criteria with respect to a resource upper bound (problem optimising `F1 w.r.t. F2`), the majority of related works on finding an exact resolution method relaxed the jobs' time-window constraints. The time criteria F1 is common to minimise C_{max} and the resource criteria F2 is limited by a constant upper bound (constant resource bandwidth). The exact approach is constructed by two

sub-problem proposed by Józefowska and Weglarz [39]. The first sub-problem is to build a job sequence called potentially optimal set (POS). Each POS is a set of feasible sequences of jobs which contain at least one sequence appearing in the optimal solution. Since the cardinality of POS grows exponentially with the number of jobs, it is hard to enumerate all the job sequence. Hence, researchers introduced many approximate methods to solve this subproblem. Jozefowska et al. [40] developed three metaheuristics to cope with the sequencing problem: the tabu search, the simulated annealing and the genetic algorithm. During the statistical test, the tabu search method out-performed the others two approaches. Hence, many further works concentration on exploiting this approach for the sequencing problem [42, 43, 78, 82]. Since the optimal resource allocation for each sequence is NP-Hard, semi-optimal scheduling is solved by heuristic introduce by Jozefowska et al. [41], Waligóra [79]. Gorczyca and Janiak [28] proposed an exact approach and two heuristics to deal with the makespan optimisation `F1 w.r.t. F2`; jobs have to be preprocessed then their ready times is dynamic. Later, with the same problem, Gorczyca et al. [29] introduced the approach to solving the F2 w.r.t F1 resource levelling problem. In this work, the authors introduce a dynamic programming model. To cope with recent issues concerning the energy reasoning problem, Różycki and Węglarz [62] proposed a general method to deal with the preemptive jobs having deadlines. In a more recent work, Rozycki and Weglarz [63] accelerated the solution procedure by grouping power consumption jobs having the same characteristics. Brinkmann et al. [15] researched the parallel computing problem with continuously shared resources on many cores. All the works concerning the DSCP suppose that the total available resource is constant.

2.4.2 Parallel task scheduling problem (PTSP)

Initially, the parallel scheduling problem regards jobs which can be processed at the same time on more than one machine. The more machines process job at a time, the faster the job can finish. In the point of view of scheduling with an additional resource, machines (or operators) are a discrete and renewable resource. When the number of machines becomes vast, this latter can be seen as a continuous resource. Concerning the rigid job scheduling problem, Cieliebak et al. [18] addressed the issue of scheduling with release date and deadline on a minimum number of machines. The objective of this problem is equivalent to the

resource levelling in the case the additional resource is continuous. The authors proposed a basic heuristic named Greedy Best Fit with the conception of the *"shiftable interval"*. The idea of the heuristic is to avoid greedily to create new resource peak while scheduling each task. The work is very inspiring, notably to our problem. The main research gaps are that the authors do not fully take into account of the sequencing of jobs and the variation of the resource. Concerning the moldable task scheduling problem, Blazewicz et al. [14] have done a research about the famous berth and quay crane allocation problem. Since the research of Blazewicz et al. [14] was based on their previous work on the DSCP, then the resource is upper bounded and the time windows constraints were ignored. The objective is also to minimise the makespan. Both the rigid and moldable jobs are typically non-preemptive. Regarding the malleable jobs, one has to take into consideration the preemption of jobs. This constraint is very crucial since it decides the complexity of the problem. A scheduling without time-windows constraints and preemptive jobs usually can be solved in polynomial times [13]. The nature of the malleable jobs is to take advantage of the available resource to speed up and complete the task as soon as possible. Then, in many works, the objective of the scheduling is to avail as much as possible the resource and optimise the production, problem optimising F1 w.r.t. F2. Based again on DSCP, Blazewicz et al. [12] introduced an exact method to solve the malleable task scheduling problem to minimise the makespan. Newly, Beaumont and Marchal [9] developed a normal form for the preemptive tasks to minimise the weighted mean completion time. The property was later formulated in [10]. The authors stated that an optimal resource allocation for each job this problem in non-decreasing in the case of the constant total available resource. Jansen and Zhang [37] address the same problem, yet taking into account of the precedence constraints. Karbasioun et al. [44] addressed the energy reasoning problem with malleable consumption tasks.

2.4.3 Cumulative Scheduling Problems (CSP)

While the DSCP and the PTSP cope with continuous time problems, Baptiste et al. [7] dealt with the discrete time problem. The discrete time model gives us the advantage to introduce the time-varying resources, yet it increases at the same time the computational effort. The scheduling problem of Baptiste et al. [7] considered both the release date and deadline for the tasks. The problem studied is a special case of the malleable task scheduling

problem where the tasks are non-preemptive, and the additional resource is single and continuous. The variation of the resource overtimes even causes the feasibility test problem to be NP-Hard to solve. To cope with this problem, Carlier and Pinson [16] introduced an adjustment of heads algorithm based on Jackson's pseudo-preemptive scheduling. We resume in table 2.2 the classification of the concerned works in this subsection with their corresponding constraints and objective, as well as the solution procedure.

2.5 Research positioning

The existed works on the EVCC resumed in table 2.1 underline a lack of the resolution methods with the time criteria optimisation. Furthermore, a complete and global scheduling which considers all the possible constraints has not been developed yet. In section 2.4, despite a significant number of a study done on the field, there is still a void on the regarding of time-varying resource constraint. Also, according to our knowledge, there are no problem, said generic and complete, which takes into account of all the following constraints: the temporal restriction of the task, the time-dependent/constant resource profile, the semi-continuous task's resource consumption. To fill the research gap, we tackle in this thesis a class of scheduling problems with additional resources with time criteria optimisation (total weighted completion times). Also, the problem takes into consideration the time-dependent resource profile and the time availability of jobs. Furthermore, jobs' resource consumption can be constant or variable. Hence, we will present in the next chapter the formal definition of this class of problems. An exact solution approach and an approximative solution approach to solving the most general problem in the class will be mentioned after that.

2.6 Conclusion

In this chapter, we studied the state-of-the-art of different subjects and themes that will be developed in the thesis. First, we presented the fundamental definitions and terminologies of the EVCC problem, then the scheduling problem with the additional resource; then

Approach	Method	Works	Constant resource	Time-varying resource	Release date	Deadline	Objective
Discrete-continuous scheduling problem	Tabu search	[78]	x				Makespan minimization
	Dynamic programming and heuristic	[65]	x				Total weighted flowtime
	Polynomial-time algorithm	[67]	x			x	the weighted number of tardy jobs/due date assignment/ makespan/ total resource consumption costs.
Cumulative Scheduling problem	Time-bound adjustment algorithms	[7]	x		x	x	Satisfiability tests
	Adjustments of heads (Jackson's pseudo-preemptive schedule)	[16]	x		x		Makespan minimization
Rigid task scheduling problem	Greedy Best Fit algorithm	[18]	x				Minimise number of machines
Moldable task scheduling problem	Suboptimal algorithm	[14]	x				Makespan minimization
Malleable task scheduling problem	Dominant class of schedules	[64]	x				Total weighted completion time
	Rectangle packing algorithm	[13]	x				Makespan minimization

Table 2.2 – Classification of some featured approaches on the theoretical scheduling problem under resource constraint

we address the optimisation criteria, i.e. how can evaluate the performance of an optimisation. The lack of existed works on the charging scheduling of electric vehicle have pushed us to study the theoretical problem of scheduling under additional resource, notably the discrete continuous scheduling, the parallel task scheduling and the cumulative scheduling problems. The literature reviews stated the research gap and also orientated the precise position of our research.

Since our problematic considers many constraints simultaneously: the variation of the total amount of available resource, the temporal restrictions, the malleable/moldable jobs, we have to begin the research with a formal definition of a new class of scheduling problem. Hence, in the next chapter, we present the EVCC problem classification and definition according to each set of constraints, followed by formal mathematical formulations.

The work done in this chapter is subject to one publication [55].

Chapter 3

EVCC problem classification and mathematical formulation

Contents

3.1 Introduction 48

3.2 EVCC problem classification 48

3.2.1 EVCC constraints 49

3.2.2 ACPF-ACPV configurations 51

3.3 Complexity analysis 54

3.3.1 Notation and complexity 55

3.4 MILP formulation 56

3.4.1 Cumulative formulation 59

3.4.2 Disjunctive formulation 62

3.4.3 LP-Relaxation analysis 66

3.5 Computational study 68

3.5.1 Random instance generator 68

3.5.2 Numerical experiments 70

3.6 Conclusion 75

3.1 Introduction

In the previous chapter, we have pointed out the necessity of defining a new class of scheduling problems. This demand is due to the lack of a complete and generic scheduling problems with single additional resource, while this resource availability can be constant or varying over time. Also, this problem has to consider all the possible constraints on the temporal availability of tasks and the way that the tasks consume the resource (moldable or malleable task)... This is the reason why in this chapter, we define firstly a new class of scheduling problems, which we name EVCC configurations. This section starts with a review on the EVCC constraints and its classification. Therefore, we present the two major derived classes of the problem according to each set of specific constraints, namely ACPF and ACPV. Since the second problem is the most generic and complete one, we decided to study the mathematical formulations of this later. We dedicated a part of this chapter to notate and study the complexity of the problem. The two formulations, one cumulative and one disjunctive, are studied in two aspects: the polytopes analysis and the computational analysis. We conclude the chapter by the numerical tests of those two formulations.

3.2 EVCC problem classification

With the literature review in the last chapter, one may find the two main disadvantages of the global EVCC approaches. The first one is the lack of temporal constraints and the time criteria optimisation. The temporal constraints assure the satisfaction of the client and the time criteria optimisation such as the total completion time minimization that could privilege the optimisation of the quality of charging service. The second study gap of the global approaches is the lack of effective solution procedures. Furthermore, a generic model that can cope with different problems with similar constraints is still a void. Given that our industrial objective is to develop a charging solution for the residential charging service, we have two top priorities. The first priority is the feasibility of the planning according to the user behaviour, including the temporal constraints of the parking, the leaving time and the daily energy demand. The second priority is to assure the quality of the service. Therefore, every EV has to be completely charged as soon as possible. Then, the choice of using the

global approach is the good research axe. To fill in the research void of this method, we would like to formulate a family of EVCC configurations with specific given constraints and properties. With that kind of classification, one can decide which researching works can be used to solve each specific problem.

3.2.1 EVCC constraints

First, we identify three groups of constraints of the EVCC issues: the human constraints, the technical limitations and the system capacity (or resource availability) constraints. Their segregation is shown in figure 3-1.

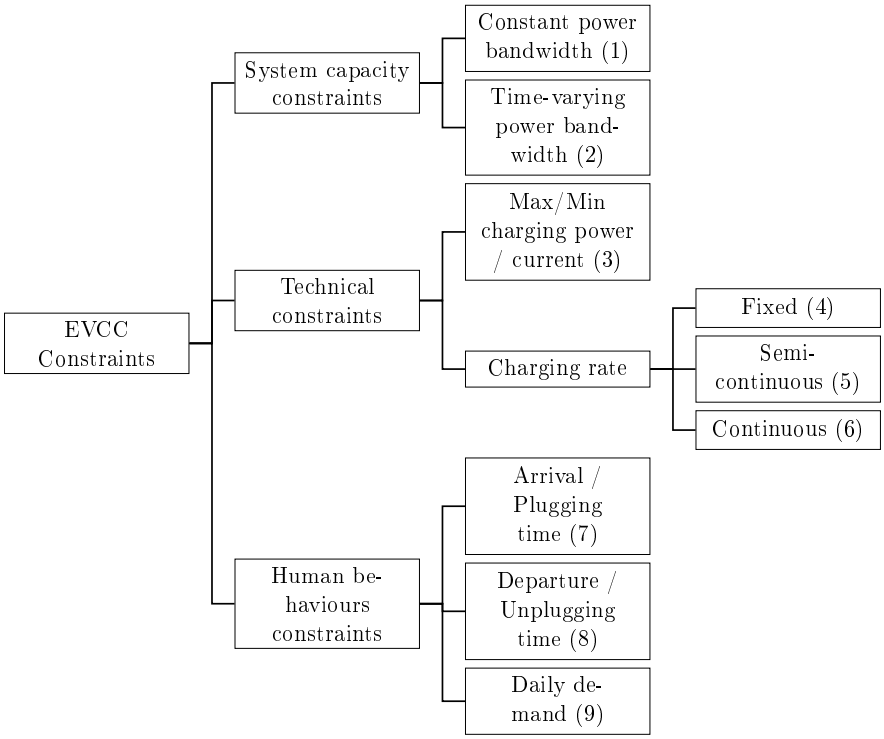


Figure 3-1 – A classification of EVCC constraints

System capacity constraints The first and also critical group of constraints is called System capacity constraints. The system capacity is the total power available for the recharging of parked EV fleet. We also denote the total available electrical power as Power Bandwidth (or Power Strip), which help us to figure out the problem geometrically as a packing problem. The power bandwidth is equivalent to the size of the single additional

resource in the theoretical scheduling problem. It is worth precision that the electric power is a continuous and renewable resource. There is two profiles of power bandwidth: the time-constant (or time independent/constant) power bandwidth profile and the time-varying (or time dependent) power bandwidth profile. The time-constant power profile is well documented in the literature, yet the counterpart is rarely mentioned. We denote a dedicated parking as a parking which is only dedicated to the charging of the EV fleet. Analogy, we denote a shared parking as a parking where the charging power is shared with other electric equipment such as the conventional ventilators or the elevators. The shared parking is very common in the professional sector where the electricity for the EV charging in the office parking is shared with industrial machines or at least computers and office devices [69]. Those two constraints are crucial not only because they change the complexity of the problem, but they also modify the way the problem can be formulated. A constant size resource problem can accept continuous time formulation whereas the time-dependent only accept continuous time formulation in the case of the resource availability is a smooth function of time. However, this assumption is ideal because, in most of the cases, the time-dependent resource fluctuates randomly. In the latter case, one should formulate the fluctuation of the resource size by a discrete time formulation.

Technical constraints The charging rate of the EV depends on the charging standard between the EVSE and the charger as we have mentioned earlier in this section. There are many charging standards, in each standard one can also find several modes and types [84]. To formulate a general and adaptive mathematical formulation, we classify the charging rate into three groups: fixed rate (constant rate), semi-continuous rate and continuous rate. To explain those constraints, we introduce some notations: let the charging job indexed by i, the scheduling horizon is divided into H intervals indexed by k. The charging rate of job i at interval k is noted by u_{ik}. Charging rate of job i is bounded by $\bar{u}_i$ and $\underline{u}_i$. If charging rate can only be fixed at the beginning of the charging process by a value $u_i^0 \in [\bar{u}_i, \underline{u}_i]$ then it is called fixed charging rate. With the parallel task problem notation, charging task with fixed rate can be considered to be *moldable*. In a special setting where $\bar{u}_i = \underline{u}_i$ the charging task is then rigid.

If at a given interval k, and the charging task has started to process: $u_{ik} \in [\bar{u}_i, \underline{u}_i]$

the charging rate is called semi-continuous or continuous depending on the lower bound $\bar{u}_i$. If $\bar{u}_i > 0$ then the charging rate is called semi-continuous and the charging task is non-preemptive. On the contrary, If $\bar{u}_i = 0$ then the charging rate is called continuous, the charging task is then preemptive.

User behaviour constraints The user behaviours consist of the parking time (or plugging time) and the departure time (or unplugging time). The plugging time is equivalent to the release date of the charging task and the unplugging time corresponds to its deadline. The behaviours of EV user defines the temporal constraints of the EVCC problem. In addition, the load demand varies with the behaviours, which is proportional to the daily travelled distance. This conduct consists the workload of the charging task. There are many ways that those constraints can be embedded in the problem. In a stochastic environment, the parking time and the daily demand can be formulated by random distribution. Usually, the unplugging times are parameters of the problem which can be decided by EV user before the charging scheduling. In the deterministic environment, the charger communicates with the charge point about its energy requirement, and the EV user fixed the charging starting time before the coordination is taking place.

3.2.2 ACPF-ACPV configurations

The segregation of the charging rate defines our two major configurations *Charging algorithm with fixed power (ACPF)* and *Charging algorithm with variable power (ACPV)*. In both of the settings, the user behaviours set of constraints is the same; the different sub-configurations are created due to the existence of other restrictions.

ACPF configurations The name of the configuration is based on the French acronym Algorithme de Charge de Puissance Fixe which means Charging Algorithm with Fixed Power. Initially, the name was created for only one algorithm. In the same way, ACPV is also initially created as a name of an algorithm. However, when the research has grown, we found that the problem ACPF is actually a set of configurations that can be treated by many different algorithms that we will present in the following chapters. Respecting the standard

that had been defined earlier for the industrial implementation, we kept on naming the two configurations ACPF and ACPV. The ACPV configuration corresponds to the EVCC constraint (4) in figure 3-1 where the charging rate is constant. The sub-configurations of ACPF: ACPF 1 and ACPF 2 correspond to the power bandwidth profile. Precisely, the constant power bandwidth defines the ACPF 1, and the time-dependent power profile defines the ACPF2. The complexity of the ACPF 1 is the same as the complexity nature of the rigid jobs scheduling problem with time windows constraints, and it is proven to be NP-Hard to find feasible solutions [18] and the makespan minimisation [18]. For the ACPF 2, there is no available research on the domain. However, adding the variation of the resource cannot make any problem simpler, the ACPF 2 is at least NP-Hard for the feasibility test and the makespan minimisation.

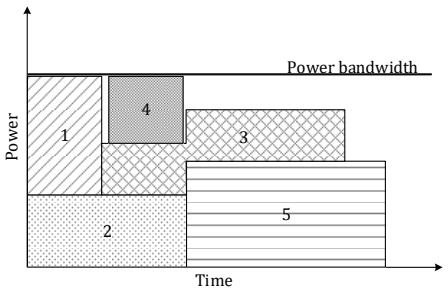
ACPV configurations The name of the configuration is based on the French acronym Algorithme de Charge de Puissance Variable which mean Charging Algorithm with Varying Power. The ACPV configuration corresponds to the constraints (5) and (6) in figure 3-1 with the charging rate are continuous and semi-continuous. With the constant power bandwidth, we have the ACPV 1 and with the time-varying power bandwidth, we have the configuration ACPV 2. Concerning the ACPV 1, we distinguish two sub configurations, ACPV 1a, which corresponds to the continuous charging rate, and ACPV 1b, which corresponds to the continuous charging rate. The ACPV 1a is the only configuration that can be solved in a polynomial time for the feasibility test problem. Regarding the ACPV 2, we tend to make the most general configuration, so the charging rate is semi-continuous and the time-dependent power profile. The complexity of the feasibility test of ACPV 1b and ACPV 2 are both NP-Hard. For time-criteria optimisation problem, both the configurations are NP-Hard in strong sense. The ACPV 2 is the most general configuration because other configurations can be deduced from ACPV 2 by relaxing one or many constraints of ACPV 2.

In table 3.1, we resume the configuration with their respecting set of constraints. Beforehand, the complexity of each optimisation is cited. We also introduce the relevant work that can be based to solve each configuration in the last column. More details on the proof of complexity of each configuration will be detailed in the following chapters. Figure

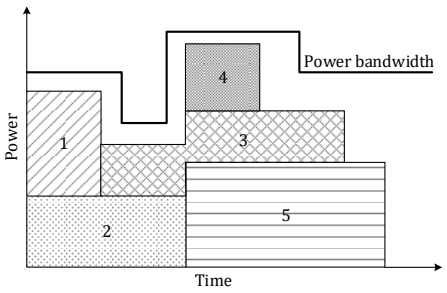
Configurations	Sub-Configurations	Power bandwidth		Charging rate			Complexity			Scheduling algorithms
		Constant	Time varying	Fixed	Continuous	Semi-continuous	Feasibility	min C_{max}	min $\sum w_i C_i$	
ACPF	1	X		X			NP-Hard [18]	NP-Hard [14]	/	Moldable task scheduling [14, 21], Rectangle packing [33], Strip packing [54]
	2		X	X			/	/	/	/
ACPV	1a	X			X		Polynomial [10]	Polynomial [13]	Polynomial [10]	Preemtable malleable task scheduling[10, 13], Discrete-continuous scheduling [39]
	1b	X				X	NP-Complete (Strong) [7]	NP-Hard (Strong) [16]		Malleable task scheduling [10, 64], Cumulative scheduling [7], Jackson pseudo-preemptive scheduling [16]
	2		X			X	NP-Hard [56]	NP-Hard (Strong) [56, 57]		Discrete malleable task scheduling [52, 56]

Table 3.1 – The ACPF - ACPV configurations

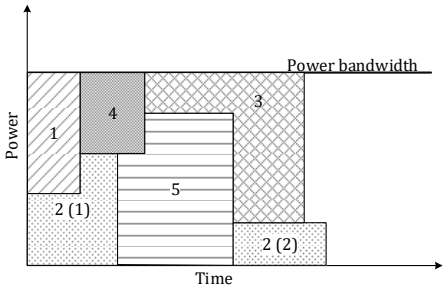
3-2 illustrates graphically the five configurations.



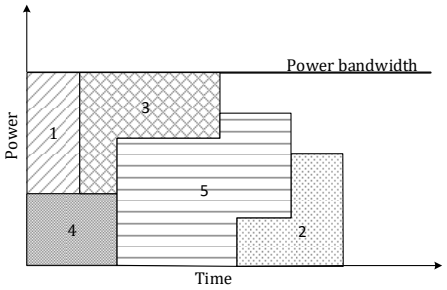
(a) ACPF 1



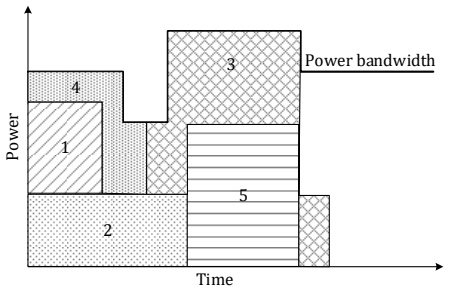
(b) ACPF 2



(c) ACPV 1a with charging task 2 being preemtable : task 2 (1) and task 2 (2)



(d) ACPV 1b - non-preemtable tasks



(e) ACPV 2

Figure 3-2 – Illustrations of 5 EVCC configurations with corresponding possible solutions

3.3 Complexity analysis

First, we would recall the ACPV configuration specifications. The ACPV is an EVCC configuration with variable charging rates. The variable charging rate can be continuous (ACPV 1a) or semi-continuous (ACPV 1b and ACPV 2). The configuration ACPV 1 is

distinguished with ACPV 2 by the power bandwidth profile. ACPV 1 has the constant power bandwidth, and ACPV 2 has the time-dependent power bandwidth. Since the ACPV 1a problem can be solved in a polynomial time and there are numerous solution procedures existing in the literature that can treat the problem, we focus in this part of the ACPV 2 configuration which is the most general configuration. Relaxing the time-varying resource constraint, one can obtain the ACPV 1b configuration from the ACPV 2. The objective chosen of the ACPV 2 problem is the total weighted completion time minimisation.

3.3.1 Notation and complexity

We use the three field notation introduced by Graham et al. [30] accompanied by the expansion of the additional resource [11] (see 2.3.3 for more details) to denote our problem. The machine environment is the identical parallel machines. Tasks have release dates and deadlines. There is only one additional resource and the resource consumption function is linear due to the accumulation of electrical power, which could be detailed later on the formulation explication. The total power is varying over times and denoted by U_k with k indexing the time's intervals. The problem notation is thus $P_m|res1\dot{:}, lin, \sum u_{i,k} \leq U_k, r_i, d_i| \sum w_i C_i$.

Regarding the complexity, Yalaoui and Chu [83] pointed out that the parallel machine scheduling problem with release date and total completion times is NP-Hard. Furthermore, if the objective is to minimise total weighted completion times, the problem is NP-Hard in strong sense [50]. Jedrzejowicz and Skakovski [38] prove that the discrete continuous scheduling problem is at least NP-Hard because the existence of the additional resource cannot make the problem any simpler than classical parallel machine scheduling problem. The ACPV 2 configuration takes into account both the time-windows constraints and the time-dependent resource size. It is thus more general than the DSCP. Hence, it is at least NP-Hard in the strong sense.

3.4 MILP formulation

The ACPV 2 problem consists of the time-dependent power bandwidth profile. The time-varying power profile has at each time interval a different continuous value of the total available power. Because of this profile, the time formulation of the problem has to be discrete. The scheduling horizon contains H time intervals (or decision intervals) indexed by k. The decision of ACPV 2 is expressed in two aspects: time decision and resource allocation. Concretely, for each job, one has to decide when can it start and end; at every interval, how many resources can this job consume. For that reason, if one wish to formulate the problem by the linear programming formulation, this latter should take the form of a Mixed Integer Linear Program with the time-indexed formulation.

We use a standard approach to formulate the ACPV 2 problem. Given a set of n jobs, $J = \{J_1, J_2, \ldots, J_n\}$ to be scheduled on a scheduling horizon of H decision intervals, each of them lasts Δt units of time. The notation $k \in \{0, ..., H\}$ indexes each decision interval (see figure 3-3). Each job has a release date r_i when it arrives into the system, and a deadline d_i, when it has to leave: $r_i \in \{0, ..., H\}$ and $d_i \in \{0, ..., H\}$. Between two interval times $[k, k+1]$ the processing of job i takes continuously an amount of resource decided by $u_{i,k}$. The total available resource at time interval k is U_k.

State of jobs The state of a job i at a decision time k is given by $x_{i,k}$, which can be considered as the measure of the processed portion of work up to the end of the interval k during the processing of job i. A job is supposed to be completed at the beginning of intervals C_i (unknown in advance). Therefore $x_{i,C_i} = \widetilde{x_i}$ where $\widetilde{x_i}$ stands for the workload (or processing demand), which is the desired final state of job i.

The processing rate of job i is described by the difference equation (3.1) which is a transformation of the differential equation introduced by Weglarz [80]

$$\dot{x}_{i,k} = \frac{x_{i,(k+1)} - x_{i,k}}{\Delta t} = f_i(u_{i,k}) \quad \forall k = 1, ..., H \quad x_{i0} = 0 \tag{3.1}$$

Where f_i is called the processing rate function taking linear form $f_i(u_{i,k}) = h_i.u_{i,k}$

with $h_i \in (0, 1]$ is the resource consumption efficiency of job i. Thus, the workload of job i can be formulated as shown in (3.2).

$$\widetilde{x_i} = \sum_{k=0}^{C_i} f_i(u_{i,k}).\Delta t = \sum_{k=0}^{C_i} u_{i,k}.h_i.\Delta t \tag{3.2}$$

Let x_i^* denote the real processing demand of job i: (equation 3.3)

$$x_i^* = \frac{\widetilde{x_i}}{h_i} = \sum_{k=0}^{C_i} u_{i,k}.\Delta t \tag{3.3}$$

Without any loss of the generality, we suppose that $\Delta t = 1$. Hence the state of job is:

$$x_{i,k} = x_{i,k-1} + u_{i,k-1} \forall i \in \mathcal{J}, k \in \mathcal{T} : k \geq 1 \tag{3.4}$$

$$x_{i,0} = 0; u_{i,0} = 0 \forall i \in \mathcal{J} \tag{3.5}$$

Job resource constraints At any time interval, the total resource all jobs consume cannot exceed the total available resource.

$$\sum_{i=1}^{n} u_{i,k} \leq U_k \quad \forall k = 1, ..., H \tag{3.6}$$

The resource consumption is bounded then: $u_{i,k} \in [\underline{u}_i, \bar{u}_i] \cup \{0\}$. Precisely, $u_{i,k} = 0$ when a job is not in process at time k. In the opposite case, $\underline{u}_i \leq u_{i,k} \leq \bar{u}_i$. We would recall that the lower bound $\underline{u}_i$ is critical for the complexity of the problem. If $\underline{u}_i = 0$ then the resource consumption of job i is continuous, hence the job is preemtive. Otherwise the resource consumption of job i is semi-continuous and the job is non-preemptive.

Supposing that $h_i = 1$ in the example in figure 3-3, figure 3-4 shows the evolution of the state of job x_i

Let $\mathcal{J} = 1, 2, ..., n$ be the set of jobs and $\mathcal{T} = 0, 1, ..., H$. In addition, the earliest

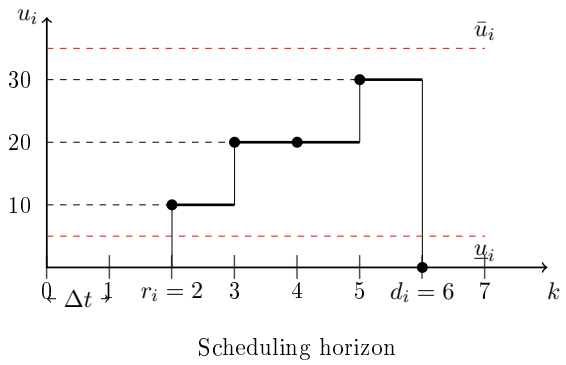


Figure 3-3 – A resource allocation example of job i

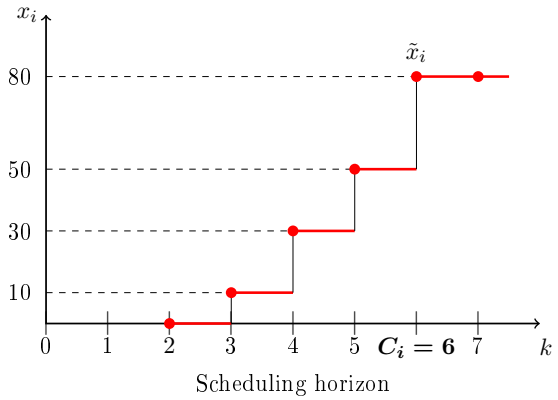


Figure 3-4 – An example of the state of job

completion time e_i and the latest starting time l_i of job i can be expressed as:

$$e_i = r_i + \lceil \frac{\tilde{x}_i}{\bar{u}_i} \rceil; \quad l_i = d_i - \lceil \frac{\tilde{x}_i}{\bar{u}_i} \rceil \quad \forall i \in \mathcal{J} \tag{3.7}$$

Therefore, the starting time of a job $i \in \mathcal{J}$ is bounded in $[r_i, l_i]$ and the completion time of this job is bounded in $[e_i, d_i]$.

Objective function ACPV 2 aims to maximize the quality of the charging service. Therefore, we set the general objective function as the total weighted completion time minimisation $\sum_{i \in \mathcal{J}} w_i C_i$.

To resume, we introduce the following variables and parameters.

Variables

- $x_{i,k}$: State of job i at decision time k
- $u_{i,k}$: Amount of resource continuously allocated to job i during interval k

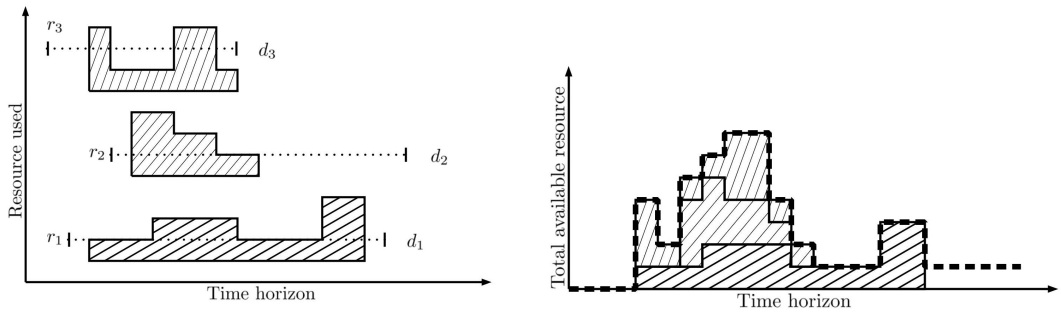
Data parameters

- $\mathcal{J}$ the set of jobs
- $\mathcal{T}$ the set of time interval
- H: scheduling horizon
- $\bar{u}_i$ and $\underline{u}_i$: maximal and minimal capacity of resource amount consumable of job i
- r_i: release time of job i
- d_i: deadline of job i
- e_i: earliest completion time of job i
- l_i: latest starting time of job i
- h_i: resource loading efficiency of job i
- $\tilde{x}_i$: resource demand of job i
- x_i^*: normalised resource demand of job i

In the next two subsections, we study the formulation of the processing indicator, the completion indicator and the semi-continuous resource consumption of jobs. Hence, we introduce two formulations named cumulative formulation and disjunctive formulation. Figure 3-5 illustrates the constraints and the resource allocations of the ACPV 2 problem.

3.4.1 Cumulative formulation

The idea that differs the two formulations is the way we formulate the processing and the completion of the jobs. In the cumulative formulation, the processing of job i is



(a) Illustration of jobs' resource allocation (b) Illustration of total time-varying resource

Figure 3-5 – The illustration of the ACPV 2 problem

noted by $y_{i,k}$. $y_{i,k} = 1$ if job i is processing at time k, otherwise, $y_{i,k} = 0$. Job completing condition is controlled by a binary variable $c_{i,k}$ which turns to 1 only if job i has already reached its final state at the beginning of interval k $x_{i,k} \geq \tilde{x}_i$; otherwise $c_{i,k} = 0$. With this given variable, the completion time of job i is:

$$C_i = H - \sum_{k=0}^{H} c_{i,k} + 1 \tag{3.8}$$

The total weighted completion time of n jobs is therefore:

$$\sum_{i \in \mathcal{J}} w_i C_i = \sum_{i \in \mathcal{J}} w_i (H + 1 - \sum_{k \in \mathcal{T}} c_{i,k}) = \sum_{i \in \mathcal{J}} w_i (H + 1) - \sum_{i \in \mathcal{J}} \sum_{k \in \mathcal{T}} w_i c_{i,k} \tag{3.9}$$

Thus, for the first model we introduce the following variables:

- $x_{i,k}$ state of job $i \in \mathcal{J}$ at time $k \in \mathcal{T}$
- $u_{i,k}$ decision variable of the amount of resource job $i \in \mathcal{J}$ consumes at time $k \in \mathcal{T}$
- $c_{i,k}$ completion indication variables, 1 if job $i \in \mathcal{J}$ already completed at time $k \in \mathcal{T}$, 0 otherwise.
- $y_{i,k}$ processing state indication variables, 1 if job $i \in \mathcal{J}$ is processing at time $k \in \mathcal{T}$, 0 otherwise.

Time-window constraints This constraint forces the processing of jobs to be between the release date and the deadline:

$$y_{i,k} = 0 \quad \forall i \in \mathcal{J}, k \in \mathcal{T} \setminus \{r_i, ..., d_i\} \tag{3.10}$$

$$y_{i,k} \in \{0, 1\} \quad \forall i \in \mathcal{J}, k \in \{r_i, ..., d_i\} \tag{3.11}$$

Semi-continuous resource consumption constraints When a job is processing at time k the resource allotted to it must less than $\bar{u}_i$ and greater than $\underline{u}_i$. Otherwise, there is zero consumption.

$$u_{i,k} \leq \bar{u}_i.y_{i,k} \quad \forall i \in \mathcal{J}, k \in \{r_i, ..., d_i\} \tag{3.12}$$

$$u_{i,k} \geq \underline{u}_i.y_{i,k} \quad \forall i \in \mathcal{J}, k \in \{r_i, ..., d_i\} \tag{3.13}$$

Non-preemption constraints Since a job has started to process and does not reach the final state, it cannot be interupted:

$$y_{i,k} \geq y_{i,k-1} - c_{i,k} \tag{3.14}$$

$$y_{i,k} \leq 1 - c_{i,k} \tag{3.15}$$

The constraints can be explained fully by the truth table 3.2. An example is shown on figure 3-6.

Table 3.2 – Constraints (3.14) and (3.15) explication

States	$y_{i,k-1}$	$c_{i,k}$	(3.14)	(3.15)	**Consequence**
Job has not started earlier	0	0	$y_{i,k} \geq 0$	$y_{i,k} \leq 1$	$y_{i,k}$ is free to be 1 or 0
Job has started earlier but it has not completed yet	1	0	$y_{i,k} \geq 1$	$y_{i,k} \leq 1$	$y_{i,k}$ must be 1, job is forced to continue on processing
Job has been already completed in this interval	0 or 1	1	$y_{i,k} \geq 0$ (or -1)	$y_{i,k} \leq 0$	$y_{i,k}$ must be 0

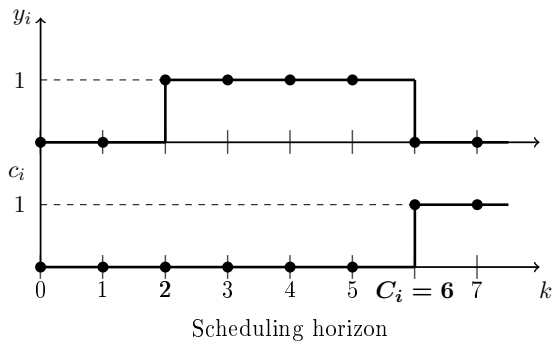


Figure 3-6 – Decision variable $y_{i,k}$ and $c_{i,k}$

Completion state There is one constraint left to model the completion state of a job. This will be:

$$x_i^* c_{i,k} \leq x_{i,k} \quad \forall i \in \mathcal{J}, k \in \mathcal{T} \tag{3.16}$$

$$c_{i,k} = 0 \quad \forall i \in \mathcal{J}, k \in \mathcal{T} : k < e_i \tag{3.17}$$

$$c_{i,k} \in \{0,1\} \quad \forall i \in \mathcal{J}, k \in \mathcal{T} : k \geq e_i \tag{3.18}$$

In equation (3.16) the completion indicator can be written as: $c_{i,k} \leq \frac{x_{i,k}}{x_i^*}$. Then is can be turn to "1" only if the state of job is greater than or equal to the workload of job i. Furthermore, the completion indicator is forced to be zero before the earliest completion time.

The cumulative model formulation: P-Cml

$$\textit{Minimize} \quad \sum_{i \in \mathcal{J}} w_i(H+1) - \sum_{i \in \mathcal{J}} \sum_{k \in \mathcal{T}} w_i c_{i,k} \tag{3.19}$$

$$\textit{s.t.} \quad (3.4 - 3.6)$$

$$(3.10 - 3.18)$$

3.4.2 Disjunctive formulation

Regarding the disjunctive formulation, we model the start and completion state of job directly by disjunctive binary variables. Those disjunctive variables belong to the SOS1

as we mentioned earlier in the beginning of the chapter.

- $\alpha_{i,k}$ job's starting state variables, 1 if job $i \in \mathcal{J}$ starts to process at time $k \in \mathcal{T}$, 0 otherwise.
- $\beta_{i,k}$ job's finishing state variables, 1 if job $i \in \mathcal{J}$ finishes at time $k \in \mathcal{T}$, 0 otherwise.

Remark 1. *For each $i \in \mathcal{J}, k \in \mathcal{T}$, the variable $y_{i,k}$ in cumulative model is equivalent to*

$$y_{i,k} = \sum_{\tau=0}^{k} (\alpha_{i,\tau} - \beta_{i,\tau}) \tag{3.20}$$

Proof. If job i is being executed at time k so $\exists \tau_1 \leq k : \alpha_{i,\tau_1} = 1$ and $\nexists \tau_2 \leq k : \beta_{i,\tau_2} = 1$ since the processing time has to be between starting time and completion time. Thus $\sum_{\tau=0}^{k} \alpha_{i,\tau} = 1$ and $\sum_{\tau=0}^{k} \beta_{i,\tau} = 0$ so by (3.20) $y_{i,k} = 1 - 0 = 1$.

In the other case, if job i has not started to process at time k yet so $\nexists \tau_1 \leq k : \alpha_{i,\tau_1} = 1$ and $\nexists \tau_2 \leq k : \beta_{i,\tau_2} = 1$ since the job has not been started yet so it cannot be ended neither. Thus, by (3.20) $y_{i,k} = 0 - 0 = 0$.

Otherwise, job i has already completed at time k so $\exists \tau_1 \leq k : \alpha_{i,\tau_1} = 1$ and $\exists \tau_2 \leq k : \beta_{i,\tau_2} = 1$ since the starting time and the completion have happened already before k. Therefore by (3.20) $y_{i,k} = 1 - 1 = 0$. □

Furthermore, the completion time of a job is now formulated as shown in equation (3.21).

$$C_i = \sum_{k \in \mathcal{T}} (k\beta_{i,k}) \quad \forall i \in \mathcal{J} \tag{3.21}$$

Time-windows constraints With starting and finishing state variables, one can formulate the time windows constraints in this way:

$$\alpha_{i,k} = 0 \quad \forall i \in \mathcal{J}, k \in \mathcal{T} \setminus \{r_i, ..., l_i\} \tag{3.22}$$

$$\alpha_{i,k} \in \{0, 1\} \quad \forall i \in \mathcal{J}, k \in \{r_i, ..., l_i\} \tag{3.23}$$

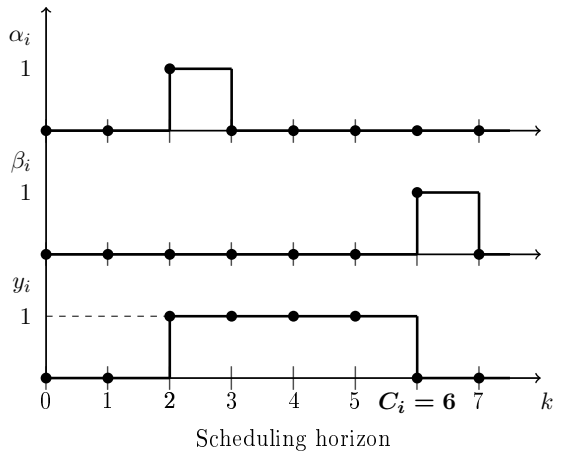


Figure 3-7 – Decision variables $\alpha_{i,k}$ and $\beta_{i,k}$ with the processing of job i

$$\beta_{i,k} = 0 \quad \forall i \in \mathcal{J}, k \in \mathcal{T} \setminus \{e_i, ..., d_i\} \tag{3.24}$$

$$\beta_{i,k} \in \{0, 1\} \quad \forall i \in \mathcal{J}, k \in \{e_i, ..., d_i\} \tag{3.25}$$

Observation 1. *With the formulations of the set of constraints, all the bounds e_i and l_i are used in $P - Dsj$ while the equivalent set of constraints in $P - Cml$ model can only use one bound e_i.*

Non-preemption constraints Since jobs are non-preemptive, only one start and one end are permitted for each job's processing. Furthermore, the finishing time must be after the starting time. Thus, the set of constraints is given by

$$\sum_{k=r_i}^{l_i} \alpha_{i,k} = 1 \quad \forall i \in \mathcal{J} \tag{3.26}$$

$$\sum_{k=e_i}^{d_i} \beta_{i,k} = 1 \quad \forall i \in \mathcal{J} \tag{3.27}$$

$$\tag{3.28}$$

With those constraints, one can find that there always one possibility for an element in α or β to be one. Therefore, the formulation is called *disjunctive*.

To reinforce the problem we introduce a cut:

$$\beta_{i,k} \leq \sum_{\tau=e_i}^{k} \alpha_{i,\tau},\ i \in \mathcal{J},\ k \in \{e_i, ..., d_i\} \tag{3.29}$$

Property 1. *Cut* $\beta_{i,k} \leq \sum_{\tau=e_i}^{k} \alpha_{i,\tau},\ i \in \mathcal{J},\ k \in \{e_i, ..., d_i\}$ *is a valid cut*

Proof. This cut naturally restricts $\beta_{i,k}$ to trigger only if there exists another $\alpha_{i,\tau}$ that is triggered where $\tau < k$. It is valid since: (i) $\sum_{r_i}^{d_i} \beta_{i,k} = \sum_{k=r_i}^{d_i} \alpha_{i,k} = 1 \leq \sum_{k=r_i}^{d_i} \sum_{\tau=d_i}^{k} \alpha_{i,\tau} = \sum_{k=r_i}^{d_i} (d_i - k)\alpha_{i,k}$ is valid and
(ii) $\sum_{\tau=r_i}^{k} (\alpha_{i,\tau} - \sum_{t=r_i}^{\tau} \alpha_{i,t}) = \sum_{\tau=r_i}^{k} (\tau + 1 - k)\alpha_{i,k} \leq 0$. Then $\sum_{\tau=r_i}^{k} (\alpha_{i,\tau} - \beta_{i,\tau}) \geq 0 \geq \sum_{\tau=r_i}^{k} (\alpha_{i,\tau} - \sum_{t=r_i}^{\tau} \alpha_{i,t})$ is also valid. □

Resource consumption constraint With remark 1, resource consumption constraints can be re-written as follows:

$$u_{i,k} \leq \bar{u}_i \sum_{\tau=0}^{k} (\alpha_{i,\tau} - \beta_{i,\tau}) \quad \forall i \in \mathcal{J}, k \in \mathcal{T} \tag{3.30}$$

$$u_{i,k} \geq \underline{u}_i \sum_{\tau=0}^{k} (\alpha_{i,\tau} - \beta_{i,\tau}) \quad \forall i \in \mathcal{J}, k \in \mathcal{T} \tag{3.31}$$

Task completion constraints The variable $\beta_{i,k}$ can only turn to 1 only if $x_{i,k} \geq \tilde{x}_i$ at a time k.

$$\beta_{i,k}.\tilde{x}_i \leq x_{i,k} \quad \forall i \in \mathcal{J}, k \in \mathcal{T} \tag{3.32}$$

The disjunctive model formulation: P-Dsj

$$\begin{aligned} \textit{Minimize} \quad & \sum_{i \in \mathcal{J}} \sum_{k \in \mathcal{T}} (w_i.k.\beta_{i,k}) \\ s.t. \quad & (3.4 - 3.6) \\ & (3.22 - 3.32) \end{aligned} \tag{3.33}$$

3.4.3 LP-Relaxation analysis

One can see that the difference between the two formulations is marked by the constraints (3.14), (3.15) vs (3.26), (3.27) and (3.29). Let us take an example to draw the feasible region of the LP-relaxation problems bounded by those constraints. We consider a scheduling problem with $H = 2$ and $n = 1$. Figures 3-8 and 3-9 show the feasible region of the variables y, c and α, β on the LP-Relaxation of the $P - Cml$ and $P - Dsj$.

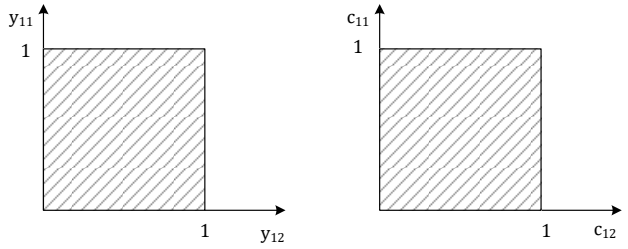


Figure 3-8 – The dashed zone is the feasible region of variables y and c in the LP-Relaxation of $P - Cml$

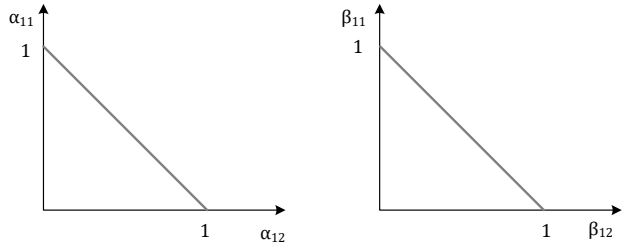


Figure 3-9 – The weighted line is the feasible region of variables α and β in the LP-Relaxation of $P - Dsj$

Figure 3-10 shows polyhedron representing the feasible space of the LP relaxation of the combination of constraints (3.14) and (3.15).

Figure 3-11 shows polyhedron representing the feasible space of the LP relaxation of the combination of constraints (3.26) and (3.29). One can notice the cut (3.29) alone is equivalent to constraint (3.14)

The hyperplanes created by the disjunctive constraints (3.26) and (3.27) strengthen the LP relaxation of $P - Dsj$. If there is no cut (3.29), it is hard to conclude which

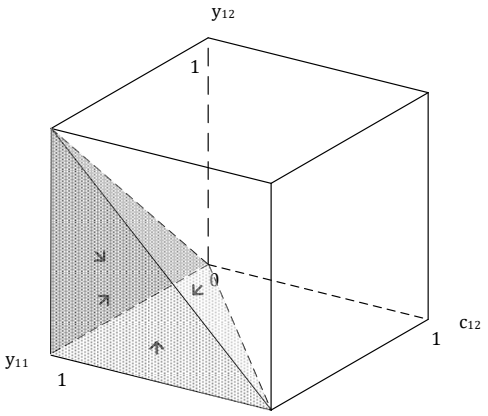


Figure 3-10 – The feasible space of the LP-Relaxation of $P - Cml$ taken from constraints (3.14) and (3.15)

formulation is stronger, since the constraints (3.14) is also a very strong constraint that reduces the searching space for $P - Cml$. We notice that the formulation can be even stronger with a simple cut:

$$\beta_{i,k} + \alpha_{i,k} \leq 1 \; i \in \mathcal{J}, \; k \in \{r_i, ..., d_i\} \tag{3.34}$$

Property 2. *Cut* (3.34) *is a valid cut*

Proof. This cut restricts that at any instance, the starting and completion of job cannot happen in the same time. We prove the validity of this cut by contradiction. Supposing that at time $k' : \alpha_{i,k'} = \beta_{i,k'} = 1$ then $\alpha_{i,k} = \beta_{i,k} = 0$ for all $k \neq k'$ then $\sum_{\tau=0}^{k}(\alpha_{i,\tau} - \beta_{i,\tau}) = 0$ for all $k \in \{r_i, \ldots, d_i\}$. For that reason, the resource consumption is zero hence job cannot be completed, it is contradicted to the supposition that job is completed at k'. □

Figure 3-12 illustrates the searching space reduction of $P - Dsj$ due to (3.34).

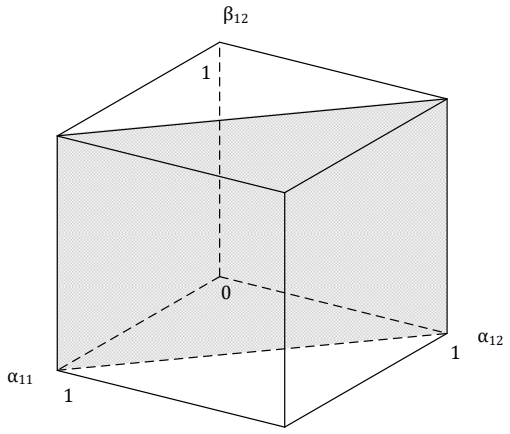


Figure 3-11 – The feasible space of the LP-Relaxation of $P - Dsj$ taken from constraints (3.26) and (3.29)

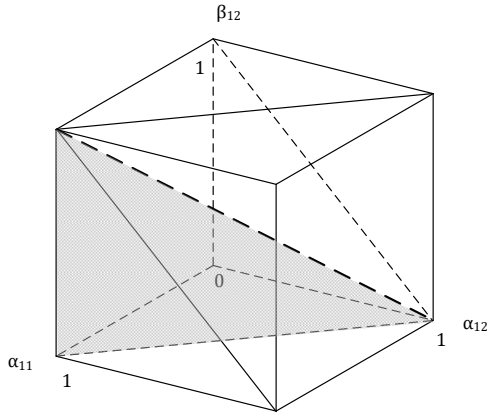


Figure 3-12 – The feasible space of the LP-Relaxation of $P - Dsj$ reinforced by cut (3.34)

.

3.5 Computational study

3.5.1 Random instance generator

Since there exists no problem which deals at the same time the time-dependent power profile and the time-windows constraints, we therefore introduce an adhoc random instance generator to study the performance of our MILP models. The way the generator create input instances is described in figure 3-13.

The set of instances is generated in the following way. The first step is the input

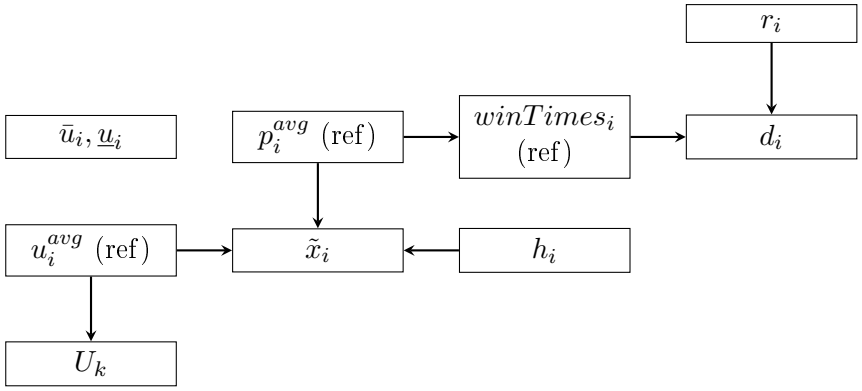


Figure 3-13 – Test instance generation

of the number of jobs n and the number of decision intervals H. Then job's release date, deadline, workload and resource consumption capacities will be randomly generated by the order given in figure 3-13. In this chart, elements which aren't pointed by an arrow will have values generated randomly and independently while the other elements have values generated based on the values of the elements in the sources of arrows pointing to them. For each job i, maximum and minimum resource consumption capacities ($\bar{u}_i$ and $\underline{u}_i$) are generated randomly by normal distribution (see table 3.3). To generate the workload (processing demand $\tilde{x}_i$) and to establish a time windows for job i, referencing elements which are average resource amount allocation u_i^{avg} and average processing time p_i^{avg} are created as reference points whose values are taken randomly according to normal and uniform law according to table 3.3. Resource consumption efficiency h_i value is generated randomly between 0.7 and 0.9. Hence, the processing demand $\tilde{x}_i$ is generated by the product of u_i^{avg}, p_i^{avg} and h_i. Release time r_i is generated from 0 to 40% of the scheduling horizon and the time windows size $winTime_i$ of each job is generated from 120% to 170% of its average processing times p_i^{avg} (which is generated previously). With that generated $winTime_i$, d_i can be generated by summing the release date and the length of window time. Finally, the total available resource amount at every moment U_k is created from 60% to 100% of the sum of all the average resource consumption of all jobs. The detailed parameters of random laws used to generate all the mentioned values can be found in table 3.3.

Table 3.3 – Random distributions parameters of generated elements for test instances

Elements	Random distribution	Parameters
T	None	$12(hours)$
H	None	$Input$
n	None	$Input$
u_i^{avg}	Normal	$y = 90; \sigma = 0.3y$
$\underline{u}_i$	Normal	$y = 12; \sigma = 0.3y$
$\bar{u}_i$	Normal	$y = 180; \sigma = 0.4y$
p_i^{avg}	Discrete uniform	$a = \lfloor 0.4 \times H \rfloor; b = \lceil 0.7 \times H \rceil$
r_i	Uniform	$a = 0; b = \lceil 0.4 \times H \rceil$
$winTime_i$	Discrete uniform	$a = \lfloor 1.2 \times p_i^{avg} \rfloor; b = \lceil 1.7 \times p_i^{avg} \rceil$
d_i	None	$d_i = r_i + winTime_i$
h_i	Uniform	$a = 0.7; b = 0.9$
U_k	Uniform	$b = \sum_{i=1}^{n} u_i^{avg}; a = 0.6 \times b$
$\tilde{x}_i$	None	$\tilde{x}_i = u_i^{avg} \times p_i^{avg} \times h_i$

3.5.2 Numerical experiments

Experimentation settings In this section, we tend to observe the empirical behaviour of the MILP formulations to solving the ACPV 2 problem. We generate random instances according to the protocol presented previously. The number of tasks is accordingly 10, 15, 20 and 30. Corresponding to each number of tasks, we generate six instances with increasing size $H \in \{20, 40, 60, 80, 100, 150, 200\}$. The instances with increasing sizes are chosen only to test the limit of the solving of the MILP by solver CPLEX. We implement the random generator in Matlab 2009. There are five different models which consist of two principal formulations $P - Cml$ and $P - Djs$ with all cut includes. Also, there are three variations of $P - Djs$: the disjunctive formulation with Cut 1 (Cut (3.29)), with Cut 2 (Cut (3.34)) and without any cut which are accordingly noted $P - Djs - Cut1$, $P - Djs - Cut2$ and $P - Djs - noCut$. We use IBM ILOG CPLEX 12.6 solver to solve the MILP models. The machine used to execute all the tests has a CPU Intel Core i5 3.20 GHz with 8GB RAM, running Linux Ubuntu 14.04. We limit the execution times of CPLEX to 1800s.

Notations For each test, we log the executing time ($Time$), the number of nodes solved ($\#Nodes$) and the MILP-Gap ($MIPGap$). As we have mentioned earlier in the chapter, the solver uses the branch and bound algorithm to solve the model. The MILP Gap is calculated

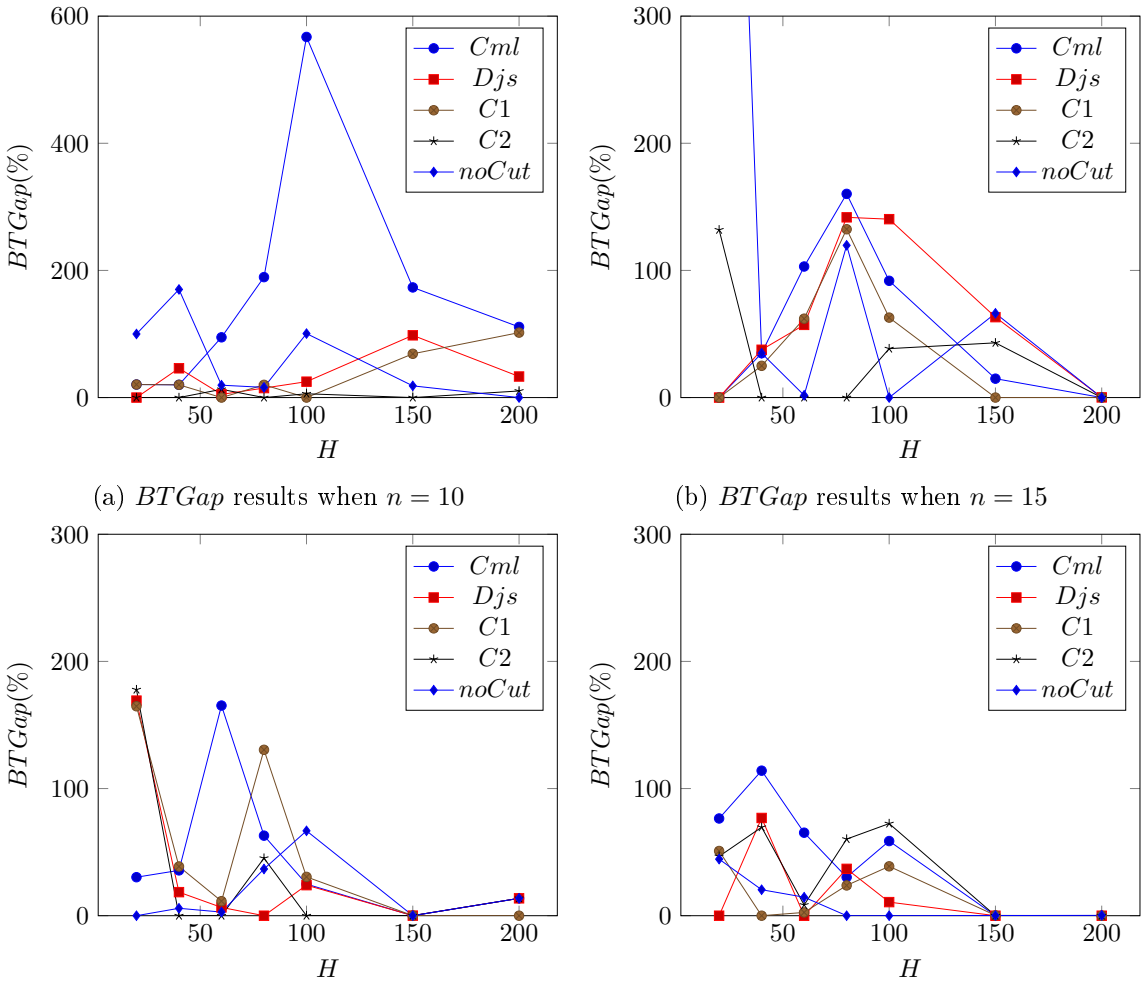
by the fraction of the difference between the incumbent upper bound and the incumbent lower bound with the upper bound: $MIPGap = \frac{UB \check{} LB}{UB}$. If the $MIPGap$ is zero, then the solution found is optimal. To have a better vision of the time spent on each instance, we normalised the difference in terms of time between the five models by the notation $BTGap$, which means Best Time Gap. For each test, let T_i be the execution time of model i. The time gap of each MILP Model is get by: $BTGap_i = \frac{T_i - \min_i T_i}{T_i}$. For example, if $BTGap_i = 300\%$ the formulation i takes 3 times longer to solve the same problem, compared to the fastest resolution time formulation. Obviously, when a method has the $BTGap = 0\%$ at a given instance, it yields the best resolution time. To track more easily the results, we make the smallest values found by each category of the test in bold, and the largest values in italic. For all the tests which are solved to an optimal solution by all the five models, for each number of tasks n, the $SumOpt$ line sums all the values resulted from those tests corresponding to each number of tasks n. The $TotalOpt$ line sums all the value resulted from tests solved to optimality by all models.

Result interpretation Table 3.4 shows the results of numerical tests in terms of the execution times. Table 3.5 lists the number of nodes executed and the MILP-Gap of the tests. First, one can find the limit of finding an optimal solution within 1800 seconds, with the inputs that have a size $nH \leq 2500$. Largest case can be solved to optimality of $n = 10$ is $H = 200$, of $n = 15$ is $H = 150$, $n = 20$. Except for the case of $n = 30$ then the largest instance solved to optimality has $H = 200$. All the models have a rate of finding optimality 24/32 except the $P - Djs - C2$, which has a slightly better performance with a rate of 25/32. Figure 3-14 plots the evolution of the Best Time Gap with the increasing of scheduling horizon. Please note that when being compared, the method yields smaller $BTGap$ is better.

In terms of execution times, the four disjunctive models distinctly out-perform the cumulative model. In small and medium instances ($n < 30$ and $H < 150$) the $P - Djs$ is at least two times faster than the $P - Cml$. This result confirms the convex analysis we have made earlier, that the disjunctive formulation is theoretically tighter than its counterpart. Observing the four variations of the disjunctive formulations: with two cuts, with cut 1, with cut 2 and without a cut, one can see the $P - Djs - Cut2$ is the most effective to

Table 3.4 – Solving times and best solving times gap resulted from different formulations

n	H	P-Cml		P-Djs		P-Djs-C1		P-Djs-C2		P-Djs-noCut	
		Time	BTGap	Time	BTGap	Time	BTGap	Time	BTGap	Time	BTGap
10	**20**	0.1	21%	**0.1**	**0%**	0.1	21%	**0.1**	0%	*0.2*	*100%*
	40	1.9	20%	2.3	46%	1.9	20%	**1.6**	**0%**	***4.2***	*170%*
	60	*16.9*	*95%*	9.1	5%	**8.7**	**0%**	9.8	13%	10.4	19%
	80	*24.0*	*189%*	9.5	15%	9.9	20%	**8.3**	**0%**	9.6	16%
	100	*208.1*	*567%*	39.0	25%	**31.2**	**0%**	33.0	6%	62.6	101%
	150	*42.6*	*173%*	30.9	98%	26.3	69%	**15.6**	**0%**	18.5	18%
	200	*73.7*	*111%*	46.5	33%	70.6	102%	38.5	10%	**34.9**	**0%**
	SumOpt	*367.2*	*244%*	137.3	28%	148.7	39%	**106.9**	**0%**	140.3	31%
15	**20**	**0.1**	**0%**	**0.1**	**0%**	**0.1**	0%	0.2	132%	*1.0*	*931%*
	40	0.8	35%	*0.9*	*38%*	0.8	25%	**0.6**	**0%**	0.8	35%
	60	*12.4*	*103%*	9.6	57%	9.9	62%	**6.1**	**0%**	6.2	2%
	80	*56.9*	*160%*	52.8	142%	50.8	132%	**21.9**	**0%**	48.0	120%
	100	46.2	92%	*57.9*	*140%*	39.2	63%	33.3	38%	**24.1**	**0%**
	150	351.6	15%	499.7	63%	**306.1**	**0%**	437.9	43%	*509.0*	*66%*
	200	1800.2	0%	1802.4	*0%*	1801.8	0%	1800.1	**0%**	1800.1	0%
	SumOpt	468.0	15%	*621.0*	*53%*	**406.9**	**0%**	500.0	23%	589.1	45%
20	**20**	0.5	30%	1.0	169%	1.0	165%	*1.0*	*178%*	**0.4**	**0%**
	40	6.2	36%	5.4	19%	*6.3*	*39%*	**4.6**	**0%**	4.8	6%
	60	*20.3*	*165%*	8.2	7%	8.5	11%	**7.7**	**0%**	7.9	3%
	80	35.3	63%	**21.7**	**0%**	*50.0*	*130%*	31.5	45%	29.7	37%
	100	173.0	25%	171.8	24%	180.8	31%	**138.6**	**0%**	*231.0*	*67%*
	150	1800.1	**0%**	1801.3	0%	1801.7	*0%*	1801.7	0%	1801.1	0%
	200	1800.3	*14%*	1800.1	14%	**1583.8**	**0%**	1800.1	14%	1800.1	14%
	SumOpt	235.3	28%	208.1	14%	246.6	35%	**183.3**	**0%**	*273.7*	*49%*
30	**20**	*1.3*	*76%*	**0.7**	**0%**	1.1	51%	1.1	47%	1.1	44%
	40	*47.3*	*114%*	39.1	77%	**22.1**	**0%**	37.5	70%	26.6	20%
	60	*21.1*	*65%*	**12.8**	***0%***	13.1	*3%*	13.9	*8%*	14.6	*15%*
	80	20.4	30%	21.5	37%	19.4	24%	*25.1*	*60%*	**15.7**	**0%**
	100	272.6	59%	190.1	11%	238.5	39%	*296.2*	*72%*	**171.8**	**0%**
	150	1800.2	**0%**	1800.1	**0%**	1804.4	***0%***	1800.1	**0%**	1802.6	**0%**
	200	1800.1	**0%**	1800.1	*0%*	1800.1	**0%**	1801.3	**0%**	1804.7	***0%***
	SumOpt	362.7	58%	264.2	15%	294.2	28%	*373.8*	*63%*	**229.8**	**0%**
	TotalOpt	*1433.3*	31%	1230.5	12%	**1096.4**	**0%**	1164.0	6%	1232.9	12%



(a) *BTGap* results when $n = 10$

(b) *BTGap* results when $n = 15$

(c) *BTGap* results when $n = 20$

(d) *BTGap* results when $n = 30$

Figure 3-14 – *BTGap* found by 5 formulations throughout different problem sizes n.

Table 3.5 – Number of nodes and MILP-Gap resulted from different formulations

n	H	P-Cml		P-Djs		P-Djs-C1		P-Djs-C2		P-Djs-noCut	
		#Nodes	MIPGap	#Nodes	MIPGap	#Nodes	MIPGap	#Nodes	MIPGap	#Nodes	MIPGap
10	**20**	***0***	**0.00%**	0	**0.00%**	0	**0.00%**	0	**0.00%**	0	**0.00%**
	40	**1228**	**0.00%**	3104	**0.00%**	2533	**0.00%**	2939	**0.00%**	*4156*	**0.00%**
	60	**5786**	**0.00%**	6109	**0.00%**	6739	**0.00%**	*9857*	**0.00%**	9147	**0.00%**
	80	5088	**0.00%**	4508	**0.00%**	*5094*	**0.00%**	4300	**0.00%**	**4099**	**0.00%**
	100	28783	**0.00%**	24631	**0.00%**	19286	**0.00%**	**17403**	**0.00%**	*30396*	**0.00%**
	150	**5971**	**0.00%**	*19770*	**0.00%**	14015	**0.00%**	7553	**0.00%**	10705	**0.00%**
	200	**12084**	**0.00%**	28228	**0.00%**	*29297*	**0.00%**	19601	**0.00%**	15908	**0.00%**
15	**20**	**0**	**0.00%**	0	**0.00%**	0	**0.00%**	*98*	**0.00%**	0	**0.00%**
	40	821	**0.00%**	*1258*	**0.00%**	1239	**0.00%**	**706**	**0.00%**	906	**0.00%**
	60	**3405**	**0.00%**	*10839*	**0.00%**	10115	**0.00%**	4012	**0.00%**	7845	**0.00%**
	80	**13726**	**0.00%**	35106	**0.00%**	*43955*	**0.00%**	18208	**0.00%**	37855	**0.00%**
	100	**6662**	**0.00%**	*29661*	**0.00%**	24835	**0.00%**	28643	**0.00%**	13672	**0.00%**
	150	**42460**	**0.00%**	2.10E+05	**0.00%**	1.33E+05	**0.00%**	*2.56E+05*	**0.00%**	1.89E+05	**0.00%**
	200	**28398**	**0.75%**	2.14E+05	4.76%	1.56E+05	*5.74%*	2.28E+05	0.78%	*3.82E+05*	1.95%
20	**20**	*506*	**0.00%**	**0**	**0.00%**	0	**0.00%**	0	**0.00%**	110	**0.00%**
	40	*4309*	**0.00%**	**2852**	**0.00%**	4060	**0.00%**	3911	**0.00%**	2868	**0.00%**
	60	*6454*	**0.00%**	4110	**0.00%**	3793	**0.00%**	**3551**	**0.00%**	4287	**0.00%**
	80	**5468**	**0.00%**	13978	**0.00%**	*27758*	**0.00%**	23566	**0.00%**	19317	**0.00%**
	100	**15747**	**0.00%**	70156	**0.00%**	*89844*	**0.00%**	58177	**0.00%**	89465	**0.00%**
	150	**38882**	**0.23%**	*3.31E+05*	1.55%	2.24E+05	2.41%	2.60E+05	*3.11%*	2.86E+05	1.49%
	200	**60621**	*0.25%*	4.00E+05	0.06%	*4.60E+05*	**0.00%**	4.00E+05	0.10%	3.39E+05	0.14%
30	**20**	*1070*	**0.00%**	468	**0.00%**	241	**0.00%**	**222**	**0.00%**	435	**0.00%**
	40	18055	**0.00%**	25106	**0.00%**	**17491**	**0.00%**	*27544*	**0.00%**	24301	**0.00%**
	60	4009	**0.00%**	5020	**0.00%**	**3295**	**0.00%**	*7848*	**0.00%**	5107	**0.00%**
	80	**2686**	**0.00%**	6304	**0.00%**	6038	**0.00%**	*9816*	**0.00%**	3816	**0.00%**
	100	**27502**	**0.00%**	60720	**0.00%**	1.07E+05	**0.00%**	*1.42E+05*	**0.00%**	50512	**0.00%**
	150	**27142**	**0.91%**	*1.64E+05*	3.60%	1.14E+05	3.27%	94667	*5.17%*	1.28E+05	2.73%
	200	**32799**	**0.10%**	1.53E+05	0.52%	1.52E+05	0.19%	*2.13E+05*	0.80%	1.89E+05	*1.14%*

deal with $n = 10$ and $n = 15$. The $P - Djs - Cut1$ deals best with instances where $n = 15$ and the $P - Djs - noCut$ deals best with large instances where $n = 30$. Overall, the $P - Djs - Cut - 1$ claims the best position in terms of the total solving times to find optimality. The disjunctive formulation with *Cut 2* takes the second position with little difference of 6% longer. With all cut of without cut, the two variations have the same total solving, which are both 12% longer than the best. The cumulative formulation is the worst in terms of resolution time, with a total resolution time being 31% longer than the best time. This observation underlines the fact that, adding many cuts does not necessarily improve the Branch-and-Bound performance. Adding cut is a trade-off for the resolution time: in one hand, it removes integer infeasibility, then the branching is faster; on the other hand, at each node, it takes longer time to solve the sub problem [45].

The $P - Cml$ solves all the instances with the least number of nodes generated. Observing the MILP-Gap behaviours, one can find that it yields the best MILP-Gap. Furthermore, the number of nodes is smallest when the instances are solved by $P - Cml$. It suggests that because the formulation is less tight than the disjunctive formulation, the

time stay at each node is longer, but it has the chance to find better bound. Regarding the disjunctive formulations, the $P - Djs - Cut1$ and the $P - Djs - Cut2$ have the least number of nodes created in a small instance while the disjunctive formulation without any cut yields a small number of nodes set up in large instances.

3.6 Conclusion

In this chapter, we defined a new class of scheduling problems, under the name of ACPF and ACPV. Along with the classes are the classification of specific set of constraints which are the system capacity constraints, the technical constraints and the human behaviours constraints, into each configuration. The most important constraints is the technical constraints on the charging rate, which consists of the fixed and the semi-continuous rate. The former founded the ACPF problem and the latter founded the ACPV problem.

In the second part of the chapter, we take a closer look on the most generic problem, the ACPV 2. This problem was proved to be NP-Hard in the strong sense. This problem is the central of the study of our thesis. We use two different techniques to construct those formulations, one with cumulative constraints and the other with disjunctive constraints. We analyse the polytopes description of each formulation. The disjunctive formulation gives a tighter description of the problem compared to the cumulative formulation. To conduct the numerical tests, we introduce a new test protocol with an adhoc random instance generator. The numerical results confirm our observation on the LP-relaxation analysis where the $P - Djs$ outperforms the $P - Cml$ formulation.

The formulations studied in this chapter are limited to the ACPV problem, but since this problem is the most generic one, when fixing parameters to constant values one can obtain the other configuration. For example, by fixing $\bar{u}_i = \underline{u}_i = u_i^o$ one can get the constant resource consumption, the ACPF configuration. For the next two chapters, we tend to resolve the problem more effectively. We start with an exact resolution method: a Branch-and-Price algorithm. Thence, to have a faster resolution time, we design a constructive heuristic, served as a good approximative solution procedure.

A part of the work done in this chapter is subject to one publication [51].

Chapter 4

Exact solution approach: A Branch-and-Price algorithm

Contents

4.1 Introduction 78

4.2 Preliminaries 78

4.2.1 Column generation 78

4.2.2 Branch-and-Price 79

4.3 The branch and price algorithm 81

4.3.1 Decomposition methods 81

4.3.2 MILP formulation 81

4.3.3 Local sub-problem formulation 82

4.3.4 Convexification approach 84

4.3.5 Linear relaxation of the master problem 85

4.3.6 Restricted master problem 87

4.3.7 The pricing problem 87

4.3.8 The branching scheme 88

4.4 Computational implementation 91

4.4.1 Pricing problem dual bound for earlier termination 91

4.4.2 Branching restriction enforcing on master and pricing problem . . 91

4.4.3 Initial solutions 93

4.5 Numerical tests 94
4.5.1 Test instances generation 94
4.5.2 Numerical results 94
4.6 Conclusion 97

4.1 Introduction

In this chapter, we present the Branch-and-Price technique to solve the ACPV 2 problem. The ACPV 2 require a significant number of decision variables when the size of the problem dilates, especially in the dimension of the scheduling horizon. Thus, the Branch-and-Price, which has proven to be more successful in solving huge MILP problem [8], is a suitable choice to tackle our problem. Also, the ACPV 2 problem has a decomposable nature. Most of the constraints (time-windows, resource demand, resource consumptions...) concentrate on the local problem, which means that those constraints only restrict specific jobs. There are only one global constraint which is the resource availability that aggregates all the local problems. This chapter begins with a preliminary (section 4.2) which provides some basic theoretical background and the principles of the Columns Generation algorithm and the Branch-and-Price framework. Then, we introduce in details the entire branch-and-price algorithm designed to solve the ACPV 2 problem (section 4.3 and section 4.4). In section 4.5 we conduct numerical tests to compare the performances of the different branching strategies for the branch-and-price and investigate the behaviours of the algorithm. Section 4.6 concludes the chapter with a brief resume and perspectives.

4.2 Preliminaries

4.2.1 Column generation

The column generation is an efficient algorithm to solve large LP problems. In LP problems, an optimal solution encounters in practice many non-basic variables (zero elements). Larger LP are sparser with zero elements [68]. The column generation tends then

to generate only variables that can probably improve the quality of the solution. A column is a set of basic variables. Supposing that the optimisation problem is the minimisation one, the variables can improve the solution quality is the one that yields the most negative cost to the problem. The column generation is a decentralised decision process; it consists of two problems: the sub-problem and the master problem. Alternately, the master problem is a representation of the original problem through the combination of columns. It thus has the same objective function. The technique of decomposing the linear problem into combinations of columns and resolving its associated variables is called Dantzig-Wolfe decomposition [68, 73]. The subproblem is defined to find new columns. The objective function of the subproblem is the reduced cost derived from the dual solution found from the Master Problem. The Column generation requires at least a feasible initial solution (a set of feasible columns) for the Master Problem to start its first iteration. If we did not have this information, we could not process to find the dual solution then it is unable to find new columns. At each iteration, the master problem is solved, then the new dual prices would be obtained from this later to form the objective function of the subproblem. The subproblem solves the subproblem with the set of local constraints, trying to find the solution which reduces the most the dual prices. If such solution is found, and it can give the objective a negative value (negative reduced cost) this solution can enter the basis since it also reduces the objective of the master problem. There is two common strategies of adding a new column(s) to the basis: one can add only the best one, i.e. the column resulting the most negative cost, or one can add a set of columns giving the negative cost. After adding the new column(s) to the pool, one can resolve the Master Problem, then generate new dual prices for the subproblem to find more columns. The algorithm repeats until the subproblem cannot find any more columns with negative cost. The linear programming problem is then solved to optimality. Figure 4-1 illustrates the overall Column Generation Algorithm.

4.2.2 Branch-and-Price

The branch-and-price algorithm provides a framework for the branch-and-bound algorithm to work together with the column generation. The objective is to take advantage of the column generation algorithm to solve the Integer Linear Programming or Mixed Integer Linear Programming. Practically, it is impossible to generate all columns for the master

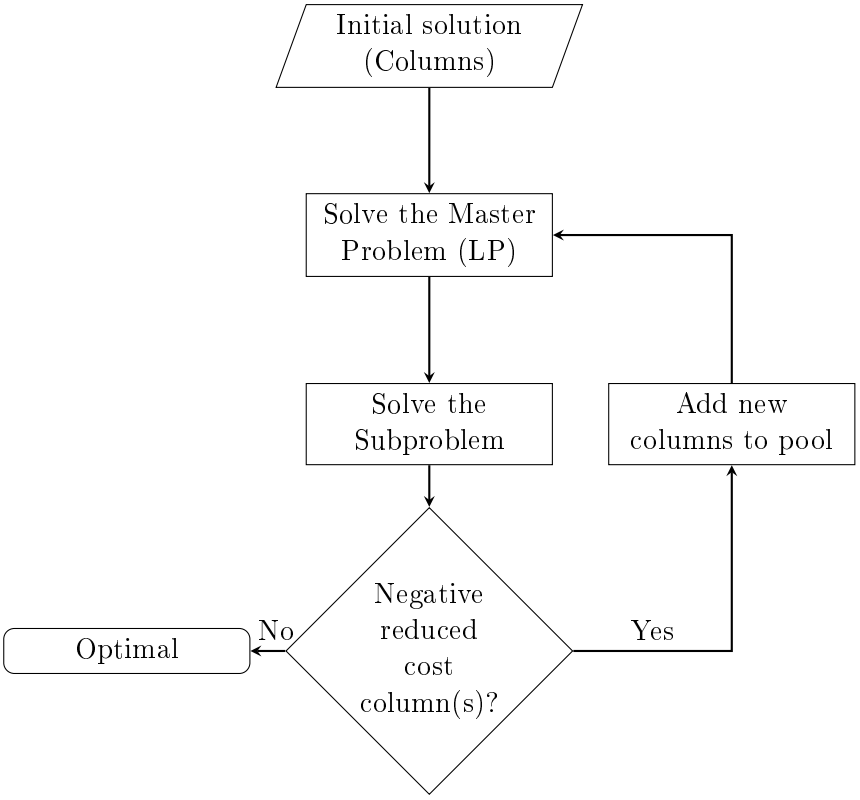


Figure 4-1 – The column generation algorithm

problem get from the MILP original problem. Thus, the Restricted Master Problem (RMP) is defined to contain a limited number of columns, called significant. The subproblem to find new column is called pricing problem since it is driven by the dual prices determined by the LP relaxation of the RMP. At the root node of the searching tree, the column generation solves the LP relaxation of the initial RMP, enlarges the first columns pool and gives the problem a LP optimal solution. If the LP-Relaxed solution is also integral, then it is the optimal solution for the MILP. Otherwise, branching occurs. At the child nodes, branching has forced some decision variables and deactivates some invalid columns. Hence it changes the dual of the RMP. At this point, the pricing problem is then resolved, and new columns are about to enter the basis. At the end of each node, the decision taking is similar to classical branch-and-bound algorithm: if integral solution found the incumbent will be updated if possible; if the LP-relaxed solution is worst than the incumbent then we prune the branch, otherwise branching happens. Elaborated and informative presentations of the algorithm can be found in [24, 27].

4.3 The branch and price algorithm

4.3.1 Decomposition methods

4.3.2 MILP formulation

In section 3, we have developed and examined two MILP models which are called $P-Cml$ and$P-Djs$. Numerical tests have shown that the former model $P-Cml$ were empirically out-performed by the latter$P-Djs$. The $P-Cml$ model is *bounded mixed integer representable* in Jeroslow's sense since it is the feasible set of continuous variable (jobs' resource allocation) and 0-1 variables (jobs' processing/completion indicators). Moreover, the$P-Djs$ is another representation of the problem by disjunctive constraints ($\alpha_{i,k} \in 0,1$ and $\sum_k \alpha_{i,k} = 1$, id. for variable β) for each job's sub-problem. First, the$P-Djs$ can exploit possible bounds according to [51]. Second, the feasible region of$P-Djs$ is a convex hull formulation of $P-Cml$ according to Jeroslow's theorem [32]. For these reasons, we choose to formulate our problem by a more compact formulation of$P-Djs$.

Original formulation *OF*

$$\text{Minimize} \qquad f(C) = \sum_{i \in \mathcal{J}} w_i C_i \tag{4.1}$$

subject to

$$\sum_{k=e_i}^{d_i} \beta_{i,k} = 1,\ i \in \mathcal{J} \tag{4.2}$$

$$\sum_{k=r_i}^{l_i} \alpha_{i,k} = 1,\ i \in \mathcal{J} \tag{4.3}$$

$$u_{i,k} \leq \bar{u}_i \sum_{\tau=r_i}^{k} (\alpha_{i,\tau} - \beta_{i,\tau}),\ i \in \mathcal{J},\ k \in \mathcal{K}_i \tag{4.4}$$

$$u_{i,k} \geq \underline{u}_i \sum_{\tau=r_i}^{k} (\alpha_{i,\tau} - \beta_{i,\tau}),\ i \in \mathcal{J},\ k \in \mathcal{K}_i \tag{4.5}$$

$$\sum_{k \in \mathcal{K}_i} u_{i,k} \geq \tilde{x}_i, \ i \in \mathcal{J} \tag{4.6}$$

$$\sum_{i \in \mathcal{J}} u_{i,k} \leq U_k, \ k \in \mathcal{K} \tag{4.7}$$

$$\alpha_{i,k}, \beta_{i,k} \in \{0,1\} \ \forall i,k \tag{4.8}$$

Constraints (4.2) and (4.3) state that each job has to start and end once. Constraint (4.4) specifies the upper bound of job i while constraint (4.5) ensures the resource consumption is above the lower bound during the processing of the job (for more details on the processing state of a job see [51]). All jobs have to be completed by accumulating enough resource by (4.6) during their corresponding time-window. At each time, the total amount the resource allotted to jobs has to be less than or equal to the system's availability U_k (4.7). The binary constraint of α and β is stated at (4.8).

4.3.3 Local sub-problem formulation

By observing the *OF*, one can group the set of constraints into local and global. Constraints (4.2)-(4.6) restrict only the value of the row vectors u_i, α_i and β_i, so they correspond to the local constraints group. All the column vectors of u are dependent on the last constraint (4.7) so it is called the global or linking constraint.

We tend to introduce a matrix representation of the local constraints to form local sub-problems. Since our LP problem is sparse because of the time-window constraints, we define some reduced row vectors, which are taken from the original row vectors of the decision variables with reduced lengths to remove zero elements.

Let $l_i = d_i - r_i$ for all $i \in \mathcal{J}$ we define:

- $u'_i \in \mathbf{R}^{l_i}$: $u'_i = \{u_{i,r_i}, u_{i,r_i+1}, \ldots, u_{i,d_i}\}$.
- $\alpha'_i \in \{0,1\}^{l_i}$: $\alpha'_i = \{\alpha_{i,r_i}, \alpha_{i,r_i+1}, \ldots, \alpha_{i,d_i}\}$.
- $\beta'_i \in \{0,1\}^{l_i}$: $\beta'_i = \{\beta_{i,r_i}, \beta_{i,r_i+1}, \ldots, \beta_{i,d_i}\}$.

By definition $C_i = \sum_{k=0}^{H} k\beta_{i,k}$ $i \in \mathcal{J}$ and $\beta_{i,k} = 0 \ \forall k \notin \{r_i, \ldots, d_i\}$. C_i can be written

as $C_i = r_i + \sum_{k=0}^{l_i-1} k\beta'_{i,k} = r_i + C'_i$. Since f is a linear function so $f(C) = f(r + C') = f(r) + f(C')$. So the minimization of $f(C)$ is equivalent to the minimization of $f(C')$ when $f(r)$ is a constant. Hence, the use of the reduced row vectors does not change the generality of our problem. For the sake of written simplicity, those reduced row vectors are called u_i, α_i and β_i when there is no ambiguity.

For the i^{th} row we can define a local sub-problem as follows:

Local sub-problem formulation for row i^{th} *LCi*

$$J_{1,l_i}\alpha_i^T = 1 \tag{4.9}$$

$$J_{1,l_i}\beta_i^T = 1 \tag{4.10}$$

$$\beta_i^T \leq \Delta_{l_i}\alpha_i^T \tag{4.11}$$

$$J_{1,l_i}u_i^T \geq \tilde{x}_i \tag{4.12}$$

$$u_i \geq \underline{u}_i\Delta_{l_i}(\alpha_i - \beta_i)^T \tag{4.13}$$

$$u_i \leq \bar{u}_i J_{1,l_i} \tag{4.14}$$

$$\alpha_i, \beta_i \in \{0,1\}^{l_i} \tag{4.15}$$

The coefficient matrices are defined as follows:

- $J_{n,m}$ is a $n \times m$ unit matrix.
- Δ_{l_i} is a lower-unit-triangular matrix with size $l_i \times l_i$: $\Delta_{i,j} = 1$ if $i \leq j$, $= 0$ *otherwise*.

Property 3. *The feasible region of LCi is bounded.*

Proof. Since the feasible region of *LCi* is a subset of the region created by (4.13)-(4.15) which is bounded, then *LCi* is bounded. □

Property 4. *The feasible region bounded by the convex hull of LCi is Minskowki representable.*

Proof. Let us consider a region *LCi'* being a relaxed region of *LCi* without constraints (4.11) and the semi-continuous constraint (4.13) becomes $u_i \geq 0$. This *LCi'* can be transformed

into Minkowski representation since the matrix formulation of constraint (4.13)-(4.15) is a unit diagonal matrix with rank $3l_i$ equal to the number of the variables, being in l_i first rows of A^{LC_i} and forming $3l_i$ linearly independent column on the matrix formulation A^{LC_i} , so $rank(A^{LC_i}) = 3l_i$ [see [68] theorem 9]. The feasible region of LCi is a subset of the region created by convex hull of LCi' is henceforth Minkowski representable. □

Now one can reduce LCi into standard LP form by adding some slack variables.

Standard LP local sub-problem formulation for row i^{th} *SLCi*

$$J_{1,l_i}\alpha_i^T - 1 = 0 \tag{4.16}$$

$$J_{1,l_i}\beta_i^T - 1 = 0 \tag{4.17}$$

$$\beta_i^T - \Delta_{l_i}\alpha_i^T + s_i^1 = 0 \tag{4.18}$$

$$\tilde{x}_i - J_{1,l_i}u_i^T + \delta_i = 0 \tag{4.19}$$

$$\underline{u}_i\Delta_{l_i}(\alpha_i - \beta_i)^T - u_i + s_i^2 = 0 \tag{4.20}$$

$$u_i - \bar{u}_i J_{1,l_i} + s_i^3 = 0 \tag{4.21}$$

with $s_i^* \in \mathbf{R}_+^{l_i}$ and $\delta_i \geq 0$.

4.3.4 Convexification approach

Classically there are two approaches to formulating the Master Problem by decomposed patterns which are discretization [71] and convexification[20, 26]. Since LCi is Minkowski representable, the convexification approach gives us the chance to fully combine two or more compatible columns to create a new feasible resource allocation. We chose this approach to formulate the master problem. Two columns are said to be compatible if they have the same vector α and β (same integral solution). The convexification of those two compatible patterns would never violate the resource utilization semi-continuous constraint. In the contrary, the solution will be infeasible and unacceptable.

Let us take a simple example to illustrate that characteristic: Let column 3 be the convex combination of column 1 and column 2. If those columns are not compatible, $\alpha^1 \neq \alpha^2$

and/or $\beta^1 \neq \beta^2$, the combination they give to α^3 and/or β^3 will be the same as resource allocation 1 or 2 ($\lambda_1 = 1$ or $\lambda_2 = 1$) to assure the feasibility. Otherwise, when they are compatible, $u^3 = \lambda^1 u^1 + \lambda^2 u^2$ will be bounded exactly in semi-continuous state: whereas $u_k^1 = 0$, u_k^2 will be also zero due to compatibility, and when $u_k^1 \in [\bar{u}, \underline{u}]$ then $u_k^2 \in [\bar{u}, \underline{u}]$. Since $\lambda^1 + \lambda^2 = 1$ (convexification constraints) $u_k^3 \in [\bar{u}, \underline{u}] \cup 0$ is then a feasible resource allocation.

Let Ω_i be the set of all extreme points of the feasible region created by *LCi*. Let $x_i^\omega = (u_i^\omega, \alpha_i^\omega, \beta_i^\omega)$ be an extreme point of Ω_i. The cost of this allocation to the original problem can be expressed as $y_i^\omega = f(C_i^\omega) + f(r_i)$ with $C_i^\omega = \sum_{k=0}^{l_i - 1} w_i k \beta_{i,k}^\omega$ Hence, *LCi* can be expressed as a convex combination:

$$u_i = \sum_{\omega \in \Omega_i} \lambda_{i\omega} u_i^\omega \tag{4.22}$$

$$\alpha_i = \sum_{\omega \in \Omega_i} \lambda_{i\omega} \alpha_i^\omega \tag{4.23}$$

$$\beta_i = \sum_{\omega \in \Omega_i} \lambda_{i\omega} \beta_i^\omega \tag{4.24}$$

$$\sum_{\omega \in \Omega_i} \lambda_{i\omega} = 1 \tag{4.25}$$

$$\lambda_{i\omega} \geq 0 \; \omega \in \Omega_i \tag{4.26}$$

$$\alpha_i, \beta_i \in \{0,1\}^{l_i} \tag{4.27}$$

With cost $y_i = \sum_{\omega \in \Omega_i} \lambda_{i\omega} y_i^\omega$.

4.3.5 Linear relaxation of the master problem

Once the integer constraints (4.15) are relaxed, one can eliminate the link between α, β and λ ($\sum_{\omega \in \Omega_i} \lambda_{i\omega} = 1$ and $\alpha_i^\omega \in \{0,1\}$ are eliminated so $\alpha_i = \sum_{\omega \in \Omega_i} \lambda_{i\omega} \alpha_i^\omega \leq 1$). The LP relaxation of the master problem can be expressed as ***MLP***:

$$\text{Minimize} \qquad z = \sum_{i \in \mathcal{J}} \sum_{\omega \in \Omega_i} \lambda_{i\omega} y_i^{\omega} \tag{4.28}$$

$$\text{subject to} \qquad \sum_{i \in \mathcal{J}} \sum_{\omega \in \Omega_i} \lambda_{i\omega} u_i^{\omega} \leq U \tag{4.29}$$

$$\sum_{\omega \in \Omega_i} \lambda_{i\omega} = 1 \qquad i \in \mathcal{J} \tag{4.30}$$

$$\lambda_i \geq 0 \qquad i \in \mathcal{J} \tag{4.31}$$

Alternatively, more simply

$$\text{Minimize} \qquad z = \sum_{i \in \mathcal{J}} \lambda_i y_i \tag{4.32}$$

$$\text{subject to} \qquad \sum_{i \in \mathcal{J}} \lambda_i u_i \leq U \tag{4.33}$$

$$1\lambda_i = 1 \qquad i \in \mathcal{J} \tag{4.34}$$

$$\lambda_i \geq 0 \qquad i \in \mathcal{J} \tag{4.35}$$

$$\lambda_i, y_i \in \mathbf{R}_+^{|\Omega_i|}$$

$$u_i \in \mathbf{R}_+^{|\Omega_i| \times H}$$

To find the right sign for dual variables, we change the constraint (4.30) to $\sum_{\omega \in \Omega_i} \lambda_{i\omega} \geq 1$ $i \in \mathcal{J}$. This change does not lose the generality of MLP since all solutions with $\sum_{\omega \in \Omega_i} \lambda_{i\omega} > 1$ $i \in \mathcal{J}$ are not optimal for the minimization problem.

Let (π, ϕ) be the pair of dual variables vectors corresponding respectively to the total resource availability constraints (or system capacity constraints) (4.29) and convexity constraints (4.30) of MLP, so $\pi \in \mathbf{R}_-^H$ and $\phi \in \mathbf{R}_+^n$. The dual program of MLP, noted DLP, is explicitly deduced from MLP formulation as follows.

$$\text{Maximize} \qquad z_D = U\pi + \sum_{i \in \mathcal{J}} \phi_i \tag{4.36}$$

$$\text{subject to} \qquad u_i^{\omega} \pi + \phi_i \leq y_{\omega i},\ i \in \mathcal{J},\ \omega \in \Omega_i \tag{4.37}$$

$$\pi_k \leq 0,\ k \in \mathcal{K}$$

$$\phi_i > 0,\ i \in \mathcal{J}$$

4.3.6 Restricted master problem

Since it is impossible to enumerate explicitly all the columns in $\Omega_i, \quad i = 1, \ldots, n$ due to its exponential size, we define a restricted set of Ω_i which is noted Ω'_i. The restricted set is thus created with a reasonable size to contain only significant columns that improve the objective value of the Master Problem. The restricted problem is enlarged by adding more columns at the pricing phase.

Let $\Omega'_i \subset \Omega_i,\ i = 1, \ldots, n$, then the restricted master problem is defined as follows ***MLP(Ω')***.

$$\text{Minimize} \qquad z = \sum_{i \in \mathcal{J}} \sum_{\omega \in \Omega'_i} \lambda_{i\omega} y_i^{\omega} \tag{4.38}$$

$$\text{subject to} \qquad \sum_{i \in \mathcal{J}} \sum_{\omega \in \Omega'_i} \lambda_{i\omega} u_i^{\omega} \leq U \tag{4.39}$$

$$\sum_{\omega \in \Omega'_i} \lambda_{i\omega} = 1 \qquad i \in \mathcal{J} \tag{4.40}$$

$$\lambda_i \geq 0 \qquad i \in \mathcal{J} \tag{4.41}$$

For all $\omega \notin \Omega'_i : \lambda_{i\omega} = 0,\ i = 1, \ldots, n$. There is a very useful property of the solution found by restricted problem proven by Feillet [24] which can be rewritten corresponding to our notation as follows.

Property 5. *[24] Let Ω' be a subset of Ω and (π^*, ϕ^*) be the optimal solution of $D(\Omega')$. If (π^*, ϕ^*) is feasible for $D(\Omega)$, (π^*, ϕ^*) is optimal for $D(\Omega)$*

4.3.7 The pricing problem

The pricing problem is introduced to price out all possible columns to decide which one can enter the basis for the next iteration of the column generation algorithm. One can deduce the reduced cost of each resource allocation by *DLP*: (4.36) and (4.37). The objective function is in minimizing sense, so the new columns wanting to enter the basis must have

the most negative reduced cost coefficient to improve the objective value. For each pricing problem i we have to find a resource allocation $\omega \in \Omega_i \backslash \Omega'_i$ such as:

$$\begin{aligned} &\text{Minimize} && c_i^* = y_{\omega i} - u_i^\omega \pi^* - \phi_i^* && (4.42) \\ &\text{subject to} && (4.9) - (4.15) \end{aligned}$$

Where π^* and ϕ_i^* is the actual optimal solution of $\mathrm{D}(\Omega')$.

There is an interesting interpretation of this pricing problem. Since $\pi^* < 0$ and according to (4.36), the most negative values of π^* are likely to be concentrated on time intervals which have little U_k to maximize the objective value of the dual problem. Those are intervals where resource bottlenecks are situated. Given that ϕ_i^* is a constant, to minimize c_i^* one has to, on one hand, reduce the weighted completion time $y_{\omega i}$ (or it can be any resource allocation costs in general). In the other hand this resource allocation probably has to take the least possible support u_i^ω at the most negative U_k. It makes a compromise between the reduced cost gained and the engagement in resource bottleneck when choosing a resource allocation: either we can reduce the objective value, or we should avoid using too much resource in the bottleneck.

4.3.8 The branching scheme

At the starting node, one initiates a feasible solution to the RMP problem by using a heuristic. The relaxation of this RMP is solved, then the dual of the LP relaxation is used to price out all the sub-problems. If column(s) with negative cost(s) is(are) found, the one(s) with most negative cost(s) will be chosen to enter the basis. Otherwise, if the optimal of the relaxation RMP is integral, then it is also optimal for the MP and the algorithm stops. In the last case, if the solution of the relaxation is not integral, and no column can be added, one has to branch the tree for new nodes. The column generation iterates again at each new node.

There are many branching strategies which are elaborated in the literature for the integer programming problem [72] such as branching on a single/many variable(s) or branching

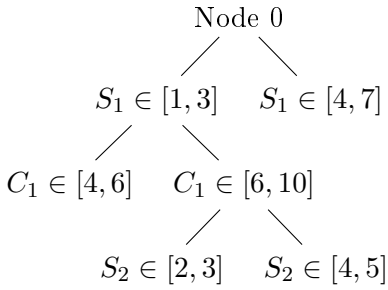


Figure 4-2 – A branching example

by adding constraints to master/pricing problem(s). Since our MIP is a *mixed 0-1 integer programming problem* and the master problem is formulated in convexification approach, it would be recommended [1, 8, 72] to branch directly on the disjunction of binary variables.

Node creation: Most(Least) Flexible SOS1 branching Given that the binary variables of each local model belong to special ordered set 1 (SOS1, i.e. at most one variable on the set can be non-zero), so according to [47] we can branch this latter into two subsets, only one can contain the non-zero variable. This division triggers two questions when dealing with our binary variable: (1) since the original problem contains n SOS1 sets (according to each variables row) which one should be chosen first and (2) when a row is chosen, how can one chose the pivot to divide the set into two subsets. To answer those two questions, we subsequently introduce the Most(Least) Flexible SOS1 branching (*MFlex* and *LFlex*) nodes creation strategy.

Since the time-windows are constrained by binary variables α and β then the branching scheme is proposed in a way that the most/least flexible job's time windows would be dichotomized to form a binary tree. An illustration is shown in figure 4-2 for the *MFlex* strategy.

The choosing of pivot to create new sub-intervals in the illustration is simply a median of the parent node available starting/completion interval. Wolsey [81] states that it would be even better if the pivot were chosen in a way that the sums of fractional variables are half-and-half on each sub-set: for $\sum_{k=1}^{k} \alpha_k = 1$ for all $\alpha_k \in S$; the division of S in to S_1 and S_2 is defined by $S_1 : \sum_{k=1}^{\kappa} \alpha_k = 1$ and $S_2 : \sum_{k=\kappa+1}^{k} \alpha_k = 1$. The pivot κ is defined as $\kappa = min\{t : \sum_{k=1}^{t} \alpha_k \geq 1/2\}$. This scheme also helps to avoid a symmetrical tree because

the fractional value changes with each parent node to create child nodes.

Algorithm 1: Most(Least) Flexible SOS1 branching

if *Found most(least) flexible interval* $S_i^* \in [s_a, s_b]$ *(or :* $C_i^* \in [C_a, C_b]$*) with* $s_b - s_a > 0$ **then**
  $\kappa = min\{t : \sum_{k=s_a}^{t} \alpha_k \geq 1/2\}$;
  Or $\kappa = min\{t : \sum_{k=C_a}^{t} \beta_k \geq 1/2\}$;
  Create node $S_i^1 \in [s_a, \kappa]$ and $S_i^2 \in [\kappa + 1, s_b]$;
  Or $C_i^1 \in [C_a, \kappa]$ and $C_i^2 \in [\kappa + 1, C_b]$;
else
  No child added
end

At each node, after the column generation phase, a decision will be taken according to 4 rules in the following order:

Pruning due to infeasibility If the Restricted Master Problem is infeasible with the new time-window restriction added on a node then this node will be pruned.

Pruning due to Bound If LP bound of the Restricted Master Problem recently solved on a node is bigger than incumbent solution $\lceil z_{RMP} \rceil \geq Incumbent$ then this node will be pruned.

Pruning due to integrality found If an integral solution is found, then we update the incumbent, this node is then pruned because, according to Property 4., this solution is optimal for the branch starting from this node.

Branching Otherwise branch for two new nodes according to Most(Least) Flexible SOS1 branching.

Node selection We propose two tree-search strategies: Best LP Bound first (BLF) and Best Projection first (BPF) [1]. In the former, the new node selected to create the child is the one with least LP bound. In the case of equality, the one with deeper level has priority (Depth-First orientation). In this approach, the projection of a node is calculated as follows:

- Sum of fractionalities $s^i = \sum_j min(f_j, 1 - f_j)$ for f_j the fractional value of the current binary variable. The sum of fractionalities of the parent node is noted s^0.

- The LP bound of RMP on node z_u^i
- The global (best) LP bound z_L
- The actual best integral bound z_{IP}

Hence, the estimated projection value at node i is defined by $E_i = z_u^i + (z_{IP} - z_L)\frac{s^i}{s^0}$. The best estimated projection selects node at $argmin\{E_i\}$.

4.4 Computational implementation

4.4.1 Pricing problem dual bound for earlier termination

To tighten the lower bound at each node, we apply the pricing dual bound introduced by Vanderbeck and Wolsey [74] for an early termination of column generation iteration to avoid the tailing off effect, also for an early pruning in the branch-and-bound tree.

$$LB_{RMP} = z^*_{RMP} + \sum_{i^* \in \mathcal{J}^*} c^*_{\omega i^*} \tag{4.43}$$

Where z^*_{RMP} is the actual LP-relaxed objective value of the restricted master problem (RMP), $c^*_{\omega i^*}$ is the reduced cost of newest columns entering the basis during the pricing phase where $\mathcal{J}^*$ is the set of sub-problems giving negative reduced cost. This bound avoids adding columns which do not improve the relaxation's bound by stopping column generation procedure when $\lceil LB_{RMP} \rceil = \lceil LB \rceil$. In the case when column generation is executed on a node, if $\lceil LB_{RMP} \rceil \geq Incumbent$, then it can be pruned.

4.4.2 Branching restriction enforcing on master and pricing problem

To tune-up the pricing phase at each node, variables in the pricing problem will be fixed according to time-window limitation of a node. The resolution of the master problem is also speeded up by deactivating columns which violate the new starting or completion interval. The deactivation of columns is done by imposing zero upper bounds on λ variables associated with those columns. The restriction on binary variables is also done by imposing

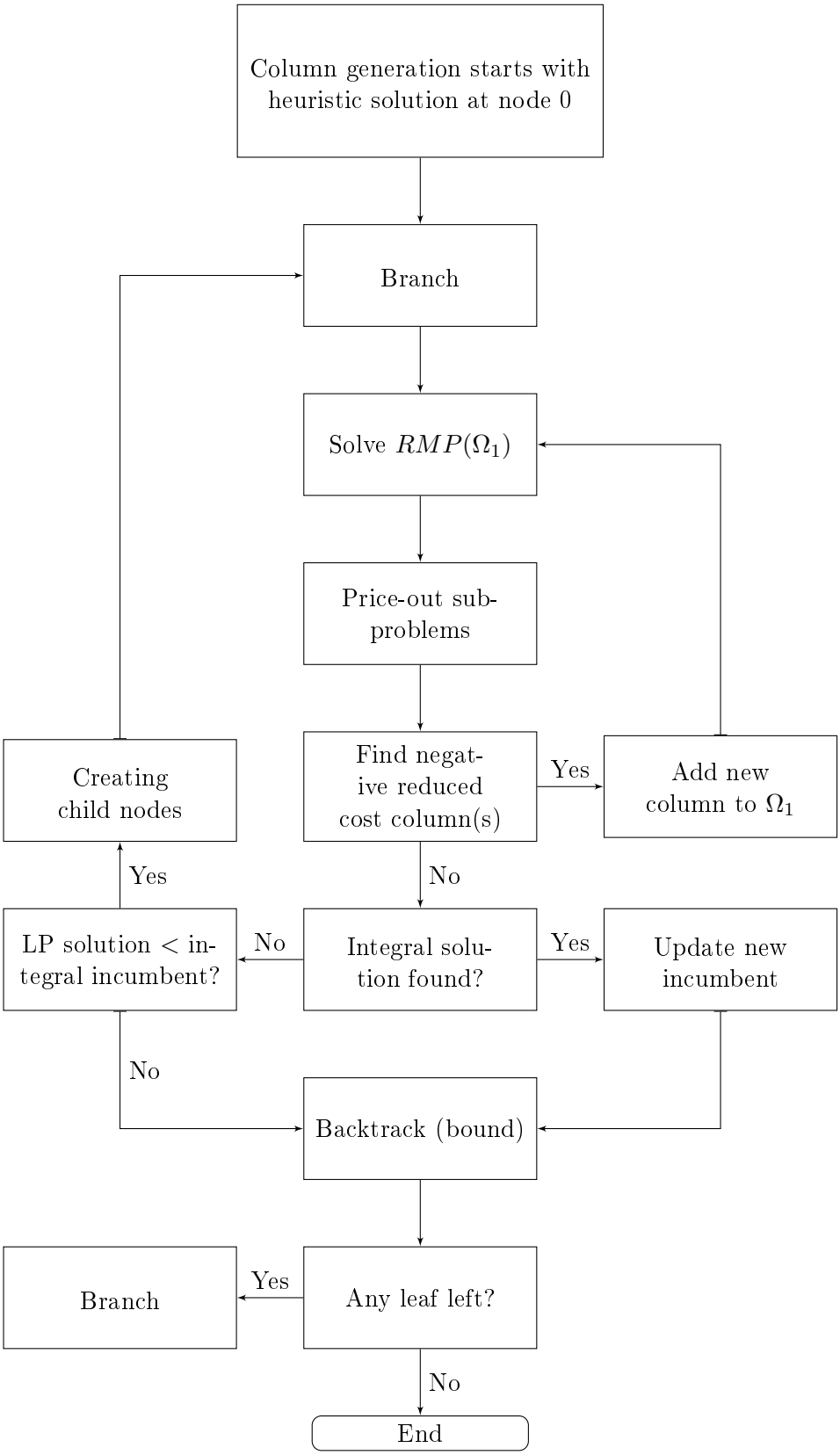


Figure 4-3 – The generic branching strategy for the ACPV 2 problem

zero bound on the set of variables which do not belong to the subset of non-zero variables defined at a node by SOS1 branching. Those columns will be reactivated, and local variables will be unfixed once the node is backtracked by resetting their bounds to $[0, 1]$.

4.4.3 Initial solutions

As proven in the last chapter, the ACPV 2 problem is NP-Hard in strong sense. Furthermore, the problem to find a feasible solution for a given instance of ACPV 2 is already NP-Hard. We propose a resource levelling algorithm to find the first solution. To begin, we present the Power peak minimisation (MinPeak) problem.

The MinPeak problem The input of the MinPeak problem consists of a set of scheduled tasks with their resource allocation and the task i to be scheduled. The objective of the scheduling is to minimise the power peak created by this new task. Let us consider a imaginary task, constructed by a reversed power bandwidth. The resource allocation of the imaginary task O is defined by $u_0 \in \mathcal{R}_+^H$, which is mathematically defined as:

$$u_{0,k} = \max_{t=0...H} U_t - U_k = U^{max} - U_k \qquad k = 0, ..., H \tag{4.44}$$

Let $\mathcal{J}'$ be the set of scheduled tasks. The MinPeak MILP problem is expressed as follows:

$$\text{Minimize} \qquad z = \max_{k \in \mathcal{K}} \left(\sum_{j \in \mathcal{J}' \cup \{0\}} u_{j,k} + u_{i,k} \right) \tag{4.45}$$

$$\text{subject to} \qquad \text{Local constraints of } LCi \tag{4.46}$$

Resource levelling algorithm The resource levelling algorithm starts with a generation of a random jobs queue. It extracts then each job in the queue and schedule this job by

solving the MinPeak MILP. The algorithm 2 details the overall process

Algorithm 2: Resource Levelling algorithm

Initialisation: Generate new random queue $\mathcal{Q}$;
foreach *Job* $i \in \mathcal{Q}$ **do**
  Solving MinPeak(i);
  if $z \leq U^{max}$ **then**
    Add u_i to $\mathcal{J}'$
  else
    Return to Initialisation
  end
end
Saving new feasible task

4.5 Numerical tests

4.5.1 Test instances generation

Data on the release dates of jobs, the deadlines, and workload, as well as the resource consumption upper-bound and lower-bound, is generated based on the random distribution parameters' database of our study case. More details on the random generation can be found in Chapter 3.

4.5.2 Numerical results

The Branh-and-Price algorithm is coded in C++ with CPLEX 12 API. The random instances generator is implemented on Matlab 2009. Tests are conducted on a Linux computer with CPU Intel Core i5 3.20GHz with 4GB RAM. We test the Branch-and-Price Algorithm with the same set of tests in chapter 3, section 3.5.2. Each instance is tested on 4 branching schemes, corresponding to the combination of two pairs of node selection: Best LP Bound First and Best Projection First, and of node creation: Most Flexible (**MFlex**) and Least Flexible (**LFlex**) SOS1 branching. With each attempt, the CPU ticks time in seconds($\#CPU$), number of total nodes ($\#Nodes$) and the number of total nodes to reach optimal solution ($\#toOpt$) is tracked. In the case that no optimal solution is found due to time-out, $\#toOpt$ denotes the numbers of nodes to reach the best solution. To track the

near-optimal trap, we log the number of nodes until the tree reaches the solution with a gap no greater than 1% (*#to1%*). The gap is calculated as $gap = \frac{z_{IP}-z_{LP}}{z_{IP}}$ where z_{IP} is the best integral solution found and z_{LP} is the least LP bound in all nodes. Execution time limit is set to *1800s*. Table 4.1 and table 4.2 show respectively the results found by the **LFlex** branching strategy and the **MFlex** branching strategy.

Table 4.1 – Numerical tests results of **LFlex branching strategy**

		Best LP-Bound Selection					**Best Projection Selection**				
n	**H**	**#to1%**	**#toOpt**	**#nodes**	**#CPU**	**gap%**	**#to1%**	**#toOpt**	**#nodes**	**#CPU**	**gap**
10	20	15	21	228	0,2	0,0%	10	10	10	0,2	0,0%
	40	730	1474	1520	4,1	0,0%	319	1420	1596	4,7	0,0%
	60	2105	4115	4785	15,7	0,0%	951	4978	5593	18,8	0,0%
	80	471	4151	4279	17,5	0,0%	1121	1808	3617	25,8	0,0%
	100	5082	13278	16393	77,3	0,0%	3090	9440	17165	63,0	0,0%
	150	2716	3717	7148	58,7	0,0%	3240	6481	9530	71,6	0,0%
	200	5142	9712	19043	134,8	0,0%	3814	13260	18164	153,2	0,0%
15	20	120	120	120	1,5	0,0%	134	156	176	0,3	0,0%
	40	353	623	929	2,1	0,0%	375	992	1103	2,1	0,0%
	60	2376	5653	8193	14,2	0,0%	3035	3763	6069	29,0	0,0%
	80	10220	29010	32966	128,0	0,0%	4712	16197	29450	86,9	0,0%
	100	7152	14464	15894	107,9	0,0%	5235	12328	16888	116,1	0,0%
	150	9171	64199	70548	419,4	0,0%	30349	66289	79866	379,6	0,0%
	200	31673	93802	121820	1800,0	5,9%	55866	57006	114011	1800,0	7,1%
20	20	20	360	1200	2,0	0,0%	34	55	72	1,4	0,0%
	40	1296	3024	3086	9,9	0,0%	345	2381	3451	18,0	0,0%
	60	385	2426	2959	22,5	0,0%	1502	2904	3338	18,8	0,0%
	80	5013	13368	23872	92,0	0,0%	4053	12158	20263	120,5	0,0%
	100	7610	34590	69180	444,9	0,0%	20215	64688	80860	457,5	0,0%
	150	68028	104659	174431	1800,0	2,9%	48438	119821	127469	1800,0	1,6%
	200	139653	170012	303593	1800,0	3,8%	47839	157869	239195	1800,0	2.9%
30	20	38	84	140	1,3	0,0%	38	113	164	2,7	0,0%
	40	3806	11299	11894	37,8	0,0%	2886	9524	9620	33,4	0,0%
	60	878	2210	2834	32,9	0,0%	684	2009	2208	38,7	0,0%
	80	1153	4210	5012	51,9	0,0%	2194	2910	4770	40,4	0,0%
	100	24826	40843	80085	653,4	0,0%	16145	59968	76882	636,7	0,0%
	150	23454	45291	80876	1800,0	2,0%	18453	41691	68346	1800,0	4,5%
	200	46651	64856	113783	1800,0	3,0%	25032	78100	100129	1800,0	4.55%

In most of the tested instances, the branching configuration *MFlex* out-performs other configurations. The first integral solutions can be found relatively rapid by the branch-and-price. During the surveillance of numerical tests, in most of the large examples, once can notice that it would take a relatively short time to find near optimal solutions. It is quick to find $gap < 1\%$, even $gap < 0.3\%$ in the majority of the cases on both of the Best-Search-First and Best-Projection-First approaches. On the other hand, *BLP-MFlex* tends to find good first solutions (which have $gap \leq 1\%$) more quickly in large scale instances. Those solutions are already useful in industrial applications where the trade-off between execution

Table 4.2 – Numerical tests results of **MFlex branching strategy**

n	H	Best LP-Bound Selection					Best Projection Selection				
		#to1%	#toOpt	#nodes	#CPU	gap%	#to1%	#toOpt	#nodes	#CPU	gap
10	20	10	10	10	**0,1**	0,0%	10	10	10	0,1	0,0%
	40	751	1521	1976	1,8	0,0%	252	729	1402	**1,7**	0,0%
	60	1031	3154	6065	**8,1**	0,0%	881	2466	2516	8,9	0,0%
	80	1564	3509	4228	10,9	0,0%	1166	1404	2649	**10,0**	0,0%
	100	1938	6461	12922	**22,8**	0,0%	1872	6151	6686	34,0	0,0%
	150	5186	9853	10371	**20,8**	0,0%	3170	4877	8129	23,4	0,0%
	200	2420	14520	17285	**53,7**	0,0%	5416	11523	11523	66,4	0,0%
15	20	21	34	34	0,2	0,0%	12	12	12	**0,1**	0,0%
	40	147	577	979	**0,7**	0,0%	266	589	719	0,8	0,0%
	60	745	4676	6777	**9,5**	0,0%	533	3836	5327	10,5	0,0%
	80	10206	27217	37801	**49,3**	0,0%	7437	11710	15824	53,3	0,0%
	100	6941	8717	16143	**35,7**	0,0%	1788	7749	14901	37,7	0,0%
	150	17451	39454	75873	**284,7**	0,0%	26249	62535	77204	297,0	0,0%
	200	63253	78715	140562	**1243,2**	0,0%	9881	73727	76008	1800,0	6,0%
20	20	17	17	17	**0,6**	0,0%	15	15	15	0,9	0,0%
	40	628	2793	3492	**6,5**	0,0%	987	1272	2192	6,6	0,0%
	60	929	2217	2996	**5,7**	0,0%	346	1557	1922	7,8	0,0%
	80	9321	11955	20263	**43,0**	0,0%	1814	10233	12954	48,5	0,0%
	100	23359	56647	58399	193,5	0,0%	19622	34716	50313	**180,8**	0,0%
	150	29698	118792	185613	1800,0	**2,0%**	20544	43459	79016	1800,0	2,5%
	200	69550	245911	248395	1663,0	0,0%	28151	128245	156397	**1599,6**	0,0%
30	20	36	101	133	**1,0**	0,0%	35	96	98	1,1	0,0%
	40	4437	8669	10320	**16,3**	0,0%	2216	6203	8862	19,7	0,0%
	60	1001	1634	2043	**11,8**	0,0%	721	1371	1757	12,5	0,0%
	80	1698	3932	4468	**11,8**	0,0%	1500	1858	3261	18,8	0,0%
	100	11885	46407	56593	233,7	0,0%	23371	29094	47695	**217,0**	0,0%
	150	13487	59848	84293	1800,0	**2,1%**	11482	30072	54677	1800,0	2,7%
	200	25245	48911	78889	1800,0	0,1%	11044	50975	84958	1800,0	0,1%

time and optimal is critical. In both of the searching strategies, the *LFlex* shows overly bad performances. While taking a closer look on SOS1 Branching, in some job configurations, the pivot is always situated at the top of the set where $\alpha_1 > 0.5$ or $\beta_1 > 0.5$. This phenomena slows down the changing of LP Bound then the branching process is trapped on branching in less insignificant intervals. With our observations, a re-ordering of binary variables according to fractional values in non-decreasing order would help to avoid this branching trap. Other observation, the larger is the test instance; the better the LP Bound which can be found after the column generation at root node. This makes the Best-Bound Search more confused since the changing of nodes causes a very less significant difference in the LP bound. It implies another way of qualifying node based on variables would be more helpful (such as best projection). This observation justifies the fact that for large instances, *BProj* gives better performances.

Compared to the results of the best method found in section 3.5.2, the **BLP-MFlex** reduces the solving time 15% to 20%.

4.6 Conclusion

This chapter presented the Branch-and-Price Algorithm to solve the ACPV 2 problem. The problem formulation is proven to be bounded and convex, which works well with the Danzig-Wolfe decomposition. The column generation algorithm approaches the problem in the convexification technique. We analyse both the primal and the dual problem. Hence we can define the pricing problem to find the new columns.

We proposed in this chapter the dual bound to stabilise the column generation at each node, which yields a good effect during the numerical tests. Concerning the branch-and-bound tree, we proposed a new branching strategy dedicated to the SOS1 set where our binary variable belongs. The numerical results underline a high sensibility of the algorithm to the branching strategy. Our **BLP-MFlex** branching strategy yields an excellent performance during the tests.

The work done in this chapter is subject to one publication [52].

Chapter 5

Approximate solution approach: Heuristics HLA

Contents

5.1 Introduction and motivation 100

5.2 Global procedure 101

5.3 Initial solution construction phase 102

5.3.1 Initial resource consumption: A reversed power bandwidth 103

5.3.2 Greedy Best Fit 104

5.3.3 Guillotine Cut and Layering 108

5.3.4 Illustrative example 109

5.4 Solution improvement phase 113

5.4.1 Greedy occupation heuristic 115

5.4.2 Shuffle heuristics 117

5.4.3 Illustrative example 119

5.5 Computational study 120

5.6 Conclusion 127

5.1 Introduction and motivation

The ACPV 2 is NP-Hard in the strong sense. Then it is hard to find exact solutions for large size instances in a reasonable time. Also, the industrial problematic requires a quick response for real life cases. A well-constructed heuristic could resolve this problem by finding a good solution more quickly: the solving time for the industrial application is demanded to be less than 20 seconds for a large parking (50 to 100) EV. Also, it could help to find an initial solution for other exact methods, especially the column generation presented in chapter 4. In addition, due to industrial demand, we cannot implement the solver CPLEX into the commercial scheduler, hence the construction of a heuristic is critical.

The EV charging scheduling problem consists of two main processes: the sequencing of jobs into the waiting queue and the allocation of the resource to each task. Local search can be a good approach to sequencing jobs. However, one cannot efficiently define a good neighbour for the continuous resource allocation. Hence, the resource allocation procedure has a more constructive nature; then the local search may not be a wise choice. This observation leads to an idea of combining a constructive method with a local search to cope with the problem. One can find two stand-out approaches in the literature that construct the solution procedure with this idea. Feo and Resende [25] introduce the conception of a GRASP (Greedy Randomized Adaptive Search Procedures). A GRASP is a multi-start heuristic. It iteratively constructs an initial solution and uses the local search to improve the quality of this latter. At every iteration, the last initial solution is completely destroyed and rebuilt a new one. The construction phase consists of a greedy constructive heuristic: the solution is constructed element-by-element in a greedy way. This phase is probabilistic because the candidate chosen is random. In the second step, the local search tends to improve the quality of the initial solution. At each iteration, only the best solution is tracked as the incumbent. There are no repair or recycle of any solution found in the previous iterations. The GRASP has proven many advantages: it is based on a simple structure and efficient to solve huge problems. Also, the construction of a greedy heuristic is handy. Nonetheless, it wastes all the solutions found on the previous iterations because the

solution destruction is total. In the case when first solutions are already hard to find, it would be very costly in terms of computational effort of the heuristic. Because of that reason, the same solution could repeatedly be found in different iterations. Thus it comes to the concept of an iterative partial destruction and reconstruction to limit those advantages which inspire Pisinger and Ropke [61] to introduce the LNS (Large Neighbour Search) approach. The LNS creates an initial solution, then for each iteration, the LNS randomly destroys only a part of the original solution and rebuild this destroyed part. The main point of that method is that one can implement different methods of destruction as well as different methods of solution reconstitution. In this point of view, the LNS is also a GRASP approach, but more configurable and adaptive. This approach inspires our heuristics introduced in this chapter.

5.2 Global procedure

This section introduces the global procedure of the heuristic HLA (Heuristic of Layering and Adapting). The HLA has many configurable versions, yet all of those contain two main phases: The construction of the first feasible solution, that we call Layering, and the solution polishing phase, that we call Adapting. As we learned in chapter 2, the feasible test of the ACPV 2 is already NP-Hard. Repeated constructions of the feasible solution are computationally expensive and wasteful. That is the reason why we use the LNS approach for our heuristic. The feasible solution is then constructed only once. At every iteration of the improvement phase, it will be partially destroyed and reconstituted. The Layering heuristic used to construct the first feasible solution for ACPV 2 is a randomised greedy constructive heuristic. The Layering heuristic is limited by a time-out while its computational time does not reach the time out and the feasible solution has not been found yet, the heuristic keeps on generating a random sequence, and according to that sequence, jobs are "cut" and "layered". If the Layering heuristic reaches its time-out without any feasible solution, the input problem is considered to be *heuristically infeasible*. Otherwise, a feasible solution enters the improvement phase, called Adapting heuristic. The adapting heuristic uses a predefined set of destruction and reconstitution methods, which conducts a large neighbour search, to improve the initial solution. During the solution polishing, any better solution caught is updated to be a new incumbent. Figure 5-1 illustrates the global

procedure of HLA

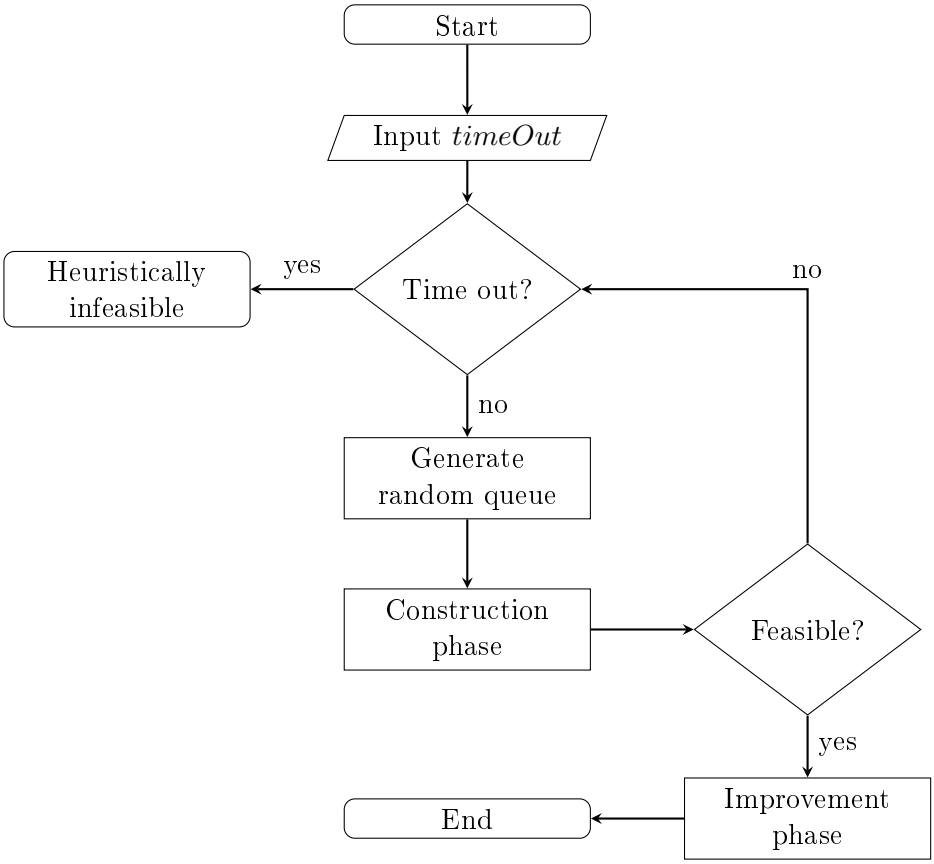


Figure 5-1 – Heuristic HLA

5.3 Initial solution construction phase

We find the inspiration to develop the feasible-solution-construction heuristic in the works related to strip packing problems. In a geometrical point of view, tasks to be scheduled on the power bandwidth is comparable to the flexible items with constant surfaces that enter a strip. In the quest of a greedy heuristic for strip packing problems, one can find many best-fit algorithms for the strip packing problems Imahori and Yagiura [34]. Conventionally, a best-fit heuristic for a feasible solution without any further objective function try is analogous with an online packing problem. Since we do not know what happens next, we try to pack the item in the position that causes the least overlap possible. Concretely, every item entering the strip and being packed in the way that it could leave the largest room possible for the following item. However, the strip packing algorithms have three main disadvantages while

facing the ACPV 2 problem. First, they do not take into consideration the time-windows constraints, i.e. packing item is free to move anywhere in the strip. Second, they suppose the strip has uniform height whereas the ACPV 2 has a varying power bandwidth. Third, the packing items usually take geometrical form (spare, rectangle, triangle, circle) when in fact, the charging task in ACPV 2 is flexible (semi-malleable). To resolve the first concern, Cieliebak et al. [18] introduced a greedy constructive heuristic to scheduling tasks on a minimal number of machines. In this work, tasks have release dates and deadlines, and it requires a fixed number of machine to process. Cieliebak et al. [18] introduce the concept of the shiftable interval to deal with the time-windows constraints. We adapt this concept and modify the original *Greedy Best-Fit* to build up resource to the ACPV 2 tasks. Before detailing the *Greedy Best-Fit* algorithm, we introduce the conception of an *Initial Resource Consumption*. It is also a key element to resolve the varying power-profile concern.

5.3.1 Initial resource consumption: A reversed power bandwidth

The original *Greedy Best Fit* algorithm can solve only instances with constant availability of machine /resource. To deal with the variation of the total resource, we present the notation of Initial Resource Consumption. This initial resource consumption can be considered as an upside-down version of U_k, noted by $U^I = \{U_0^I, ..., U_H^I\}$, which is mathematically defined as:

$$U_k^I = \max_{t=0..H} U_t - U_k = U^{max} - U_k \qquad k = 0, ..., H \tag{5.1}$$

To facilitate the presentation of the algorithms, a simple example is given throughout this chapter. In this example, we consider a scheduling problem with 5 jobs and 10 intervals long scheduling horizon whose details are shown in table 5.1 and table 5.2.

Figures 5-2a shows the time-dependent power-profile while figure 5-2b changes this latter into the intial resource consumption, which are comparable to an imaginary task already scheduled. With that initial resource consumption, the total available resource of

Table 5.1 – Properties of jobs used in example

Job i	r_i	d_i	$\underline{u}_i$	$\bar{u}_i$	$\tilde{x}_i$	h_i
1	1	9	2	11	25	0.9
2	2	10	2	13	20	0.8
3	3	10	1	6	23	0.8
4	2	9	2	12	30	0.7
5	1	8	2	8	23	0.8

Table 5.2 – Total resource amount available in example

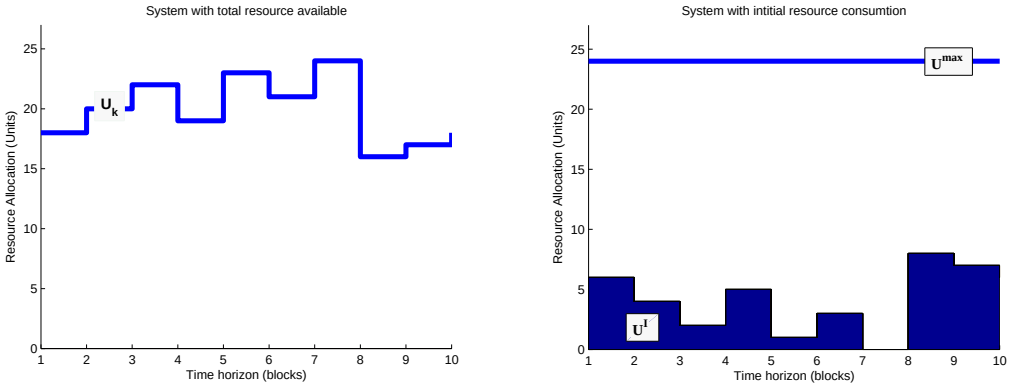
k	1	2	3	4	5	6	7	8	9	10
U_k	18	20	22	19	23	21	24	16	17	18

each interval is now considered constant and equal to U^{max}.

5.3.2 Greedy Best Fit

Shiftable interval The key of the GBF algorithm is the aggregation of job's availability and its resource consumption which is introduced under the name of "*Shiftable interval*". Each task is an imaginary piston, with two ends limited by the release date and the deadline. The diameter of the piston represents its resource consumption. Each position of the piston corresponds to a placement of task. Formally defined, a shiftable interval is a n-tuple $J_i = \langle r_i, d_i, p_i, u_i^o \rangle$, which is described according to figure 5-3 where $p_i = \lceil \frac{x_i^*}{u_i^o} \rceil$ and $u_{i,k} = \begin{cases} u_i^o \text{ if } k \in \{r_i + \phi_i, ..., r_i + \phi_i + p_i\} \\ 0 \text{ otherwise} \end{cases}$. ϕ_i is called the slack and $\phi_i^{max} = d_i - p_i - r_i$ is the maximum slack in the interval. With each ϕ_i, we have a placement of resource allocation vector, noted $\mathcal{P}_{\phi_i}$.

The idea of the GBF algorithm is to schedule without creating any new resource peak, which means the maximums total resource consumption all over the scheduling horizon. In the case that we obligate to create a new resource peak, the job's placement should be where the newly created peak is minimal. For our problem, we introduce the conception of the moldable shiftable interval, which means the shiftable interval with modulable height u_i^o. To create the least resource peak, the initial $u^o - i$ takes values of the smallest possible resource consumption $u_i^o = max\{u_i^{min}, \frac{x_i^*}{(d_i - r_i)}\}$ where $\frac{x_i^*}{(d_i - r_i)}$ is the smallest resource consumption



(a) System with total resource availability U (b) System with initial resource consumption U^I

Figure 5-2 – The initial resource consumption explication

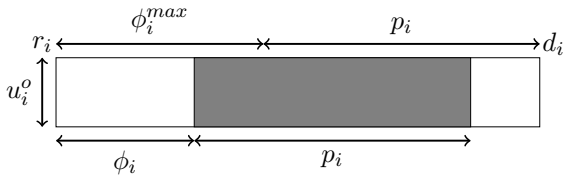


Figure 5-3 – Shiftable interval $J_i = \langle r_i, d_i, p_i \rangle$

required on each interval to ensure that job i would be completed within the time-window.

Let Ω denote the set of scheduled jobs, which includes imaginary task representing the Initial Resource Consumption U^I. Let U_Ω be the allocation of resource assigned to all those jobs. The height of an interval is the peak of resource utilisation of the jobs in Ω plus the resource utilisation of job i. For a job i, the algorithm calculates for every placement the maximum height inside the shiftable interval. From the subset of placements which lead to the lowest height, we chose the one staying in this lowest height in the least duration.

Complexity Without loss of generality, let us assume that $\phi_i^{max} = H/c; c > 0$, then the GBF$(i)$ algorithm can be computed in $O(\phi_i^{max2}) = O(H^2)$. By this complexity, one can note that, the algorithm works faster with stricter time-windows constraints.

We illustrate the algorithm with job i having a given resource allocation u_i^o in a simple example.

Figure 5-4 shows three possible placements for job i. One can find that place place-

Algorithm 3: *Greedy Best Fit* algorithm

Data: Shiftable interval $J_i = \langle r_i, d_i, p_i \rangle$
Result: Best placement $\mathcal{P}_{\phi_i^*}$
$\phi_i^{max} = d_i - r_i - p_i$;
for $\phi_i = 0..\phi_i^{max}$ **do**
  $h_{max}[\phi_i] = \max_{k \in \langle r_i+\phi_i, r_i+\phi_i+p_i \rangle} \{U_\Omega[k] + u_{i,k}\}$;
  $\Sigma[\phi_i] = \sum_{k \in \langle r_i+\phi_i, r_i+\phi_i+p_i \rangle} U_\Omega[k] + u_{i,k}$;
end
$h^* = \min_{\phi_i=0..\phi_i^{max}} h_{max}[\phi_i]$;
$\Sigma^* = \min_{\phi_i=0..\phi_i^{max}} \{\Sigma[\phi_i] | h_{max}[\phi_i] = h^*\}$;
$\phi_i^* = \min_{\phi_i=0..\phi_i^{max}} \phi_i$ wherether $\begin{cases} h_{max}[\phi_i] = h^* \\ \Sigma[\phi_i] = \Sigma^* \end{cases}$

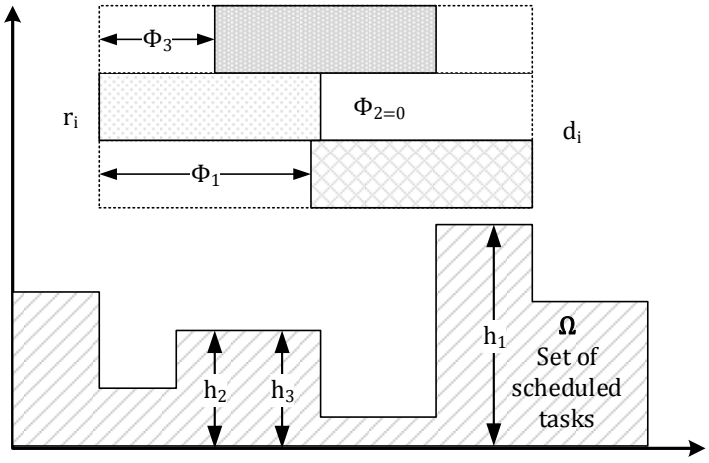


Figure 5-4 – Maximum resource heights created by three placements of job i

ment ϕ_1 has the biggest resource height, then this placement is totally not "best-fit". Both placements with ϕ_2 and ϕ_3 yield the minimal of all the maximal resource height (minimax resource height). Hence, to decide which placement is the best-fit one, we have to examine the resource height accumulation $\Sigma(\phi)$. Figure 5-5 shows that slack ϕ_2 gives a bigger resource height accumulation than the slack ϕ_3. Thus, with this given resource allocation u_i^o, the best-fit placement is with slack ϕ_3.

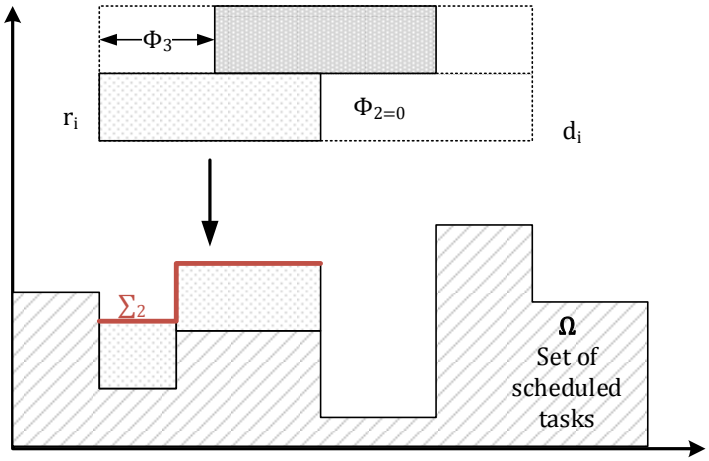


Figure 5-5 – Resource height accumulation of second placement ϕ_2

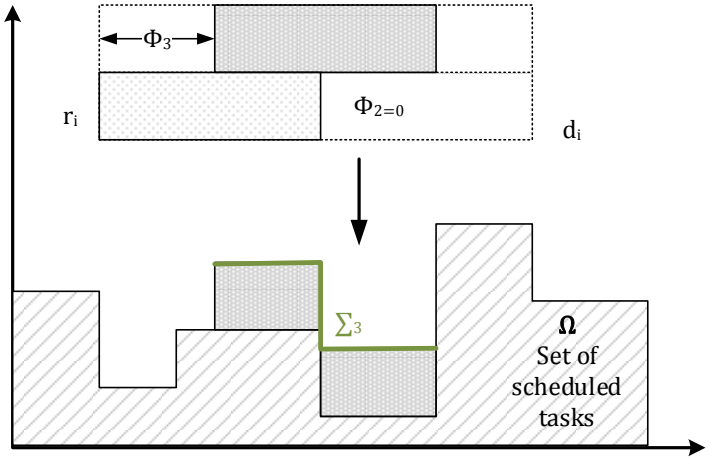


Figure 5-6 – Resource height accumulation of third placement ϕ_3

5.3.3 Guillotine Cut and Layering

Throughout empirical experiments, we observe that when the power bandwidth fluctuates sharply, it is harder to find feasible solutions. Given that the jobs' resource allocation is semi-continuous, we come to an idea that one can apply guillotine cut to flatten the resource allocation. The basic of this cut is to make the power bandwidth left as uniform as possible to avoid with a greater probability the resource consumption overlap, or we call this phenomenal simply resource bottleneck.

Each guillotine cut tentative reduces the actual minimax resource height h^* of the packing item (i.e. resource allocation for a given job). During this layering and cutting process, the temporal starting time of job is noted α'_i and the temporal completion time of job is noted β'_i. The interval $[\alpha'_i, \beta'_i]$ is called the spanning interval of job i. Algorithm 4 finds the cut point of an given job's resource allocation. For the notation, let $\hat{u}_{i,k}$ be the remained resource allocation of job i after the latest cut and let x'_i be the surface of the job's cut portion.

Algorithm 4: *Guillotine Cut finder*

Data: $u_{i,k}$, U_Ω and spanning interval $[\alpha'_i, \beta'_i]$
Result: After-cut allocation $\hat{u}_i$, job's cut portion x'_i
$gcut_1 = \min_{k \in \langle \alpha'_i, \beta'_i \rangle} U_\Omega[k] + u_{i,k}$;
$gcut_2 = \max_{k \in \langle \alpha'_i, \beta'_i \rangle} U_\Omega[k] + u_i^{min}$;
$cutPoint = max\{gcut_1, gcut_2\}$;
$\forall k \in \langle \alpha'_i, \beta'_i \rangle : cut[k] = \max\{0, U_\Omega[k] + u_{i,k} - cutPoint\}$;
$x'_i = \sum_{k \in \langle \alpha_i, \beta_i \rangle} cut[k]$;
$\forall k \in \langle \alpha_i, \beta_i \rangle : \hat{u}_{i,k} = u_{i,k} - cut[k]$;

To explain the algorithm, the guillotine cut $gcut_1$ is the minimum resource height over the newly allocation of resource with job i. The guillotine cut $gcut_2$ is the maximum height of the allocation of resource without job i over the spanning interval, plus the job's minimum resource allocation allowed. One can find that $gcut_1$ is more radical than $gcut_2$. However, $gcut_2$ assures the validity of the semi-continuous constraint of task resource consumption. It keep the resource allocation from going lower than $\underline{u}_i$. The final guillotine cut is the maximum of those previously found guillotine cuts. Figure 5-8 illustrates the two guillotine cuts.

There are two observations concerning the cutting process:

Observation 1 After each cut, the new cut portion should be smaller or equal to the previous one. If the new cut portion is as big as the old one, the result will be unchanged. Thus, this latter turns into the stop condition of the guillotine cutting iterations.

Observation 2 Since tasks are non-preemptive, from the second GBF placement, with each shiftable interval $J_i = \langle r_i, d_i, p_i \rangle$, a placement $\mathcal{P}_{\phi_i}$ is valid only if $r_i + \phi_i \leq \beta'_i$ and $r_i + \phi_i + p_i \geq \alpha'_i$.

Observation 3 The main challenge for the reallocation of resource after a cut is the task's semi-continous resource consumption and non-preemtive constraints. That is the reason why we introduce the conception of the spanning interval. if the greedy best fit happens outside the actual spanning interval of job, the minimum resource allocation is bounded as $\bar{u}_i$. Otherwise, there exist no such bound is the allocation is happened to be inside the actual spanning interval.

According to the three observations mentioned above, the *greedy best fit* algorithm combined with cutting process of a job can be described in algorithm 5. Then the overall *layering and cutting* process is described in algorithm 6.

Complexity Since algorithm 5 can be implemented in time $O(H^2)$, algorithm 6 can be computed in $O(nH^2)$.

5.3.4 Illustrative example

To illustrate the resource allocations layering process, the previously introduced example is taken.

First the resource allotted to job 1 is planned by the GBF algorithm. This allocation of resource is shown in figure 5-7.

Algorithm 5: *After-cut resource reallocation* (ACRR)

Data: After-cut allocation $\hat{u}_i$, job's cut portion x'_i, U_Ω, spanning interval $[\alpha'_i, \beta'_i]$
Result: New resource allocation u_i
$p_i^{max} = d_i - r_i$;
$p_i^{min} = \lceil \frac{x_i^*}{u_i^{max}} \rceil$;
$\delta_{i,\cdot} = 0$;
for $p'_i = p_i^{max}$ *down to* p_i^{min} **do**
  $\delta = x'_i / p'_i$;
  if $\delta \geq u_i^{min}$ **then**
    $r'_i = \max\{r_i, \alpha'_i - p'_i\}$;
    $d'_i = \min\{d_i, \beta'_i + p'_i\}$;
  else
    $r'_i = \alpha'_i$;
    $d'_i = \beta'_i$;
  end
  $GBF(r'_i, d'_i, \delta, p'_i)$;
  if *New best-fit allocation found* **then**
    Update u_i
  end
end

Algorithm 6: *Guillotine Cut and Layering Algorithm* (GC&L)

Data: Set Q of n ordered jobs
Result: Initial resource allocation U_I of n jobs
foreach $i \in Q$ **do**
  $\alpha_i = r_i$;
  $\beta_i = d_i$;
  $x'_i = x_i^*$;
  while $newPortion < oldPortion$ **do**
    $oldPortion = newPortion$;
    Placing cut portion into the strip by using algorithm 5;
    Find cut portion using algorithm 4;
    $newPortion = x'_i$;
  end
end

Figure 5-8 shows the two possible guillotine cuts for the intitial resource allocation of job 1. To avoid processing interruption, we chose to guillotine the resource allocation by $gcut_2$.

Figure 5-9 shows the resource reallocation of job 1 after being cut. The dash line represents the previous job's resource allocation. This figure also shows that $gcut_2$ can still be applied to lowering the resource peak.

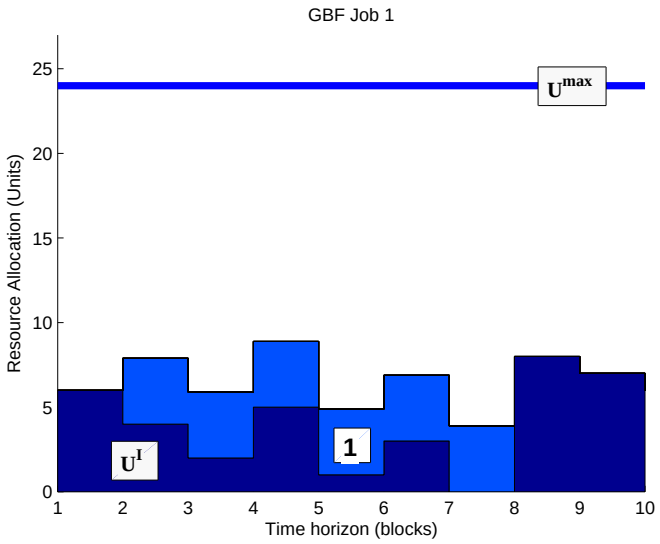


Figure 5-7 – Resource allocation of job 1 being planned by *GBF* algorithm.

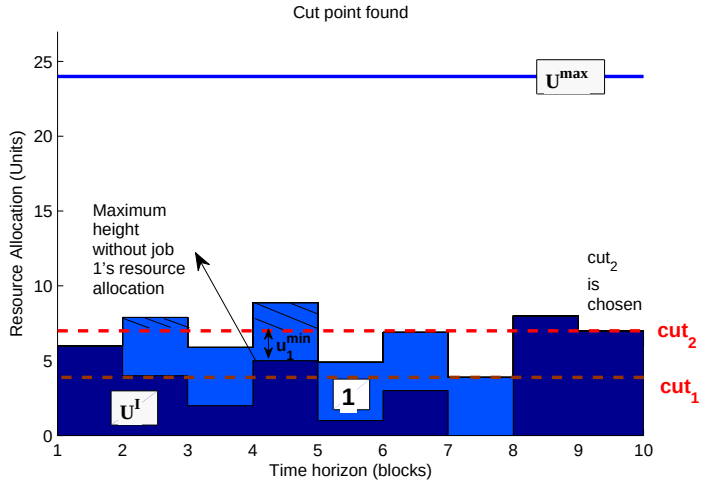


Figure 5-8 – Guillotine cuts of job 1: Initial iteration

Figure 5-10 shows the resource reallocation of job 1 after the being cut for the second time. The cutting process stops at the second iteration since there is no resource left to cut.

In the same way, job 2's resource allocation is initially found by *GBF* algorithm, then being cut by $gcut_2$ (see figure 5-11). Figure 5-12 shows the reallocation of cut resource of job 2. The cutting process then stops because there are no more resource left to cut.

Figure 5-13 shows the initial result of *GBF* on job 3. In the first cutting attempt, the $gcut_2$ is chosen then figure 5-14 shows the resource reallocation after this cut. At this time,

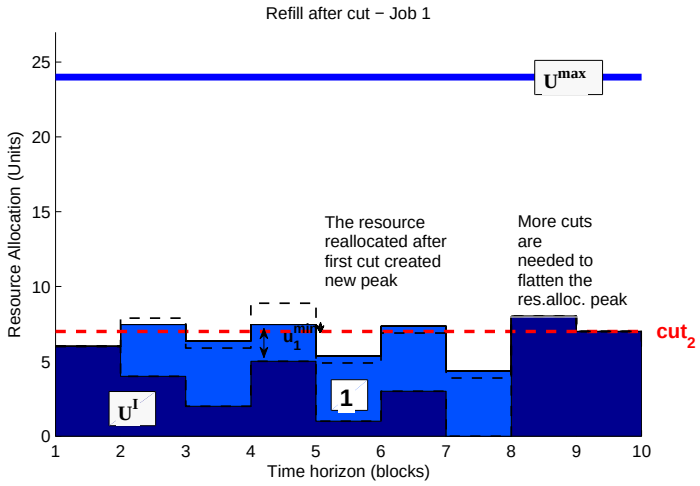


Figure 5-9 – Job 1's after-cut resource reallocation: First iteration

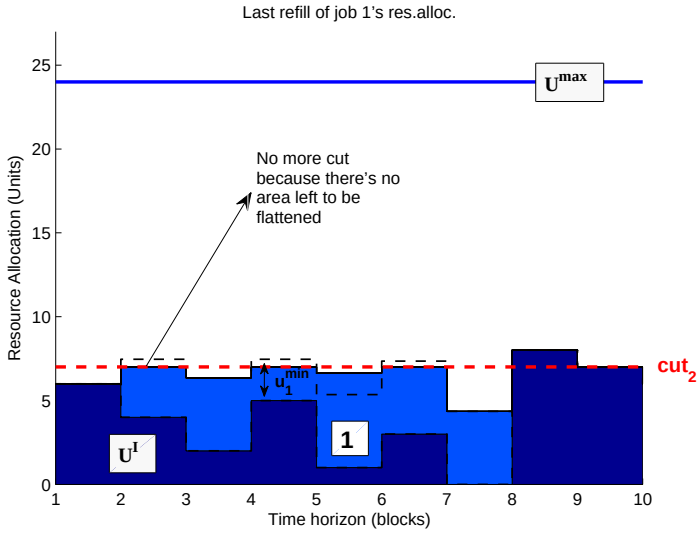


Figure 5-10 – Job 1's after-cut resource reallocation: Second iteration

$gcut_1 = \max\{gcut_1, gcut_2\}$ hence $gcut_1$ is chosen. This guillotine cut flattens completely the resource allocation hence the cutting process stops (figure 5-15).

In the same way of layering and cutting, the resource allocation of job 4 and job 5 are placed into the strip, which lead to the initial layered solution in figure 5-16.

The combination of jobs time-availability constraint and the variation of the total available resource cause the feasibility test NP-Hard [31]. An instance which can be solved by the Layering process is called heuristically feasible. On the contrary, it is heuristically

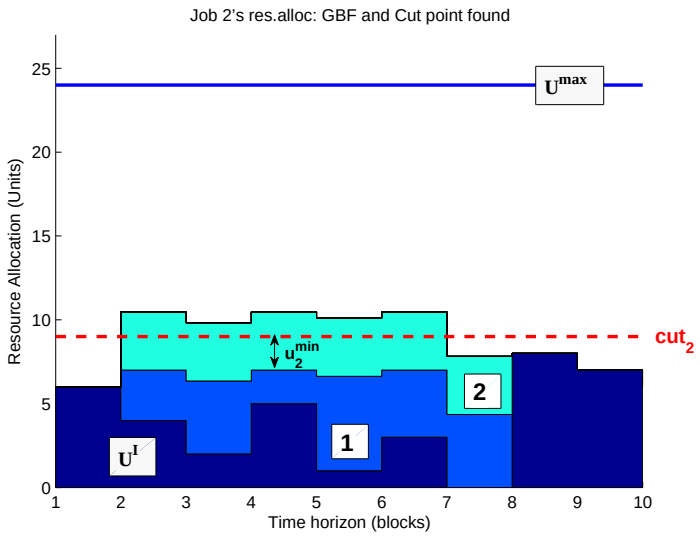


Figure 5-11 – Resource allocation of job 2 found by GBF and guillotine cut $gcut_2$

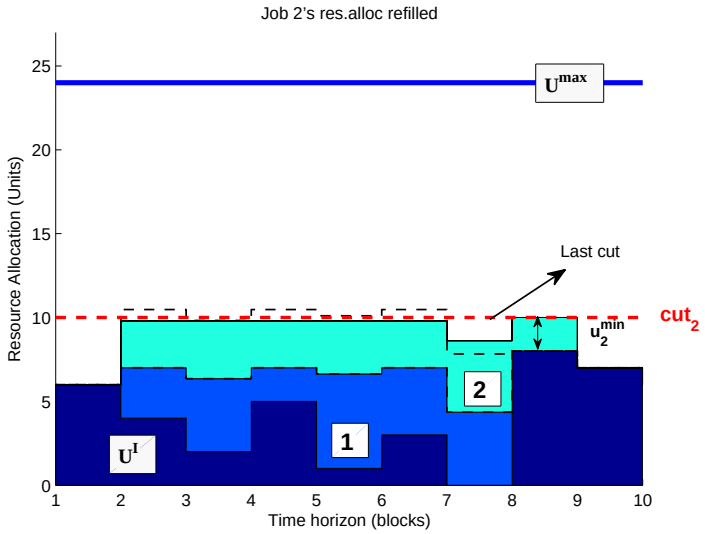


Figure 5-12 – Job 2's after-cut resource reallocation: First and also last iteration

infeasible. To ensure as much as possible the heuristic feasibility of an instance, the cut processes (cut_1 and cut_2) are introduced beside of GBF.

5.4 Solution improvement phase

This section proposes three large-neigbour search methods to improve the initial solution found by the *Layering* heuristic. This phase is called "Adapting", as the part of

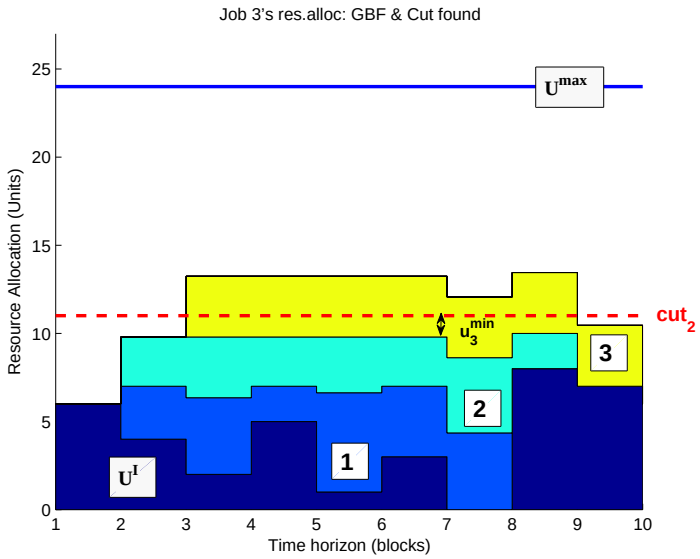


Figure 5-13 – Job 3's resource allocation found by *GB* and guillotine cuts

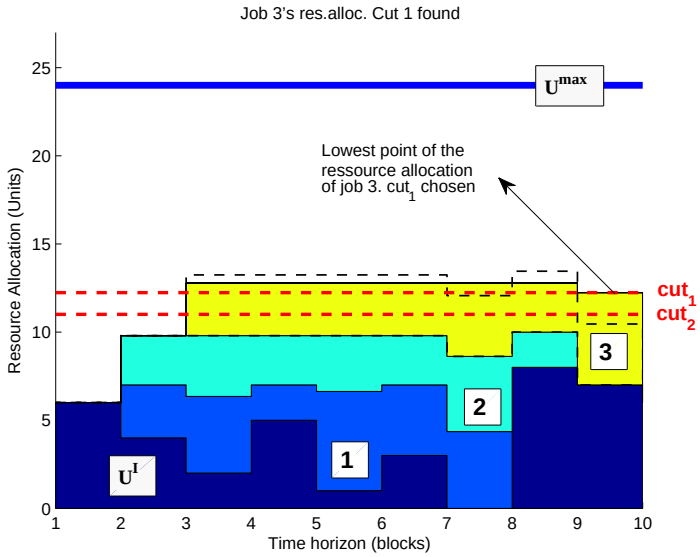


Figure 5-14 – Job 3's resource allocation: first shuffle and cut_1 found

the HLA: Heuristic of Resource layering and Adapting due to the initial conception of the second phase is to adapt the job's resource consumption to the power bandwidth variation, which is not really closed to the final conception of the heuristic. However, with respect to the naming of the heuristic, and to facilitate the industrial documentation, we decided not to change the name of the solution improvement phase.

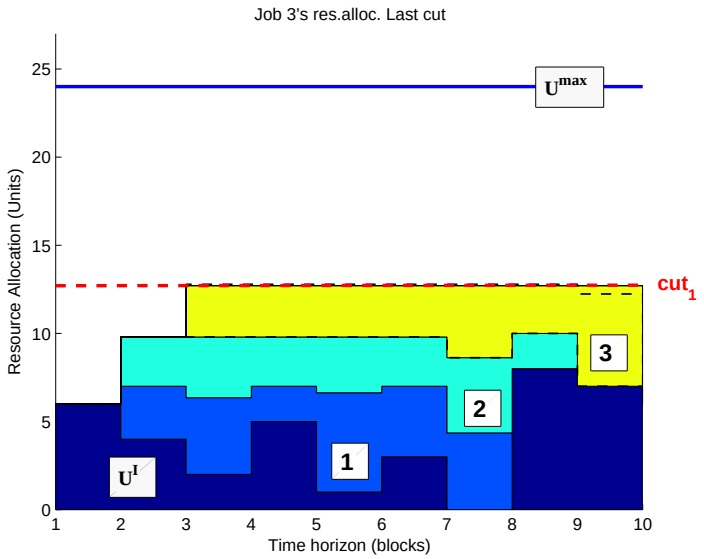


Figure 5-15 – Job 3's resource allocation: last cut

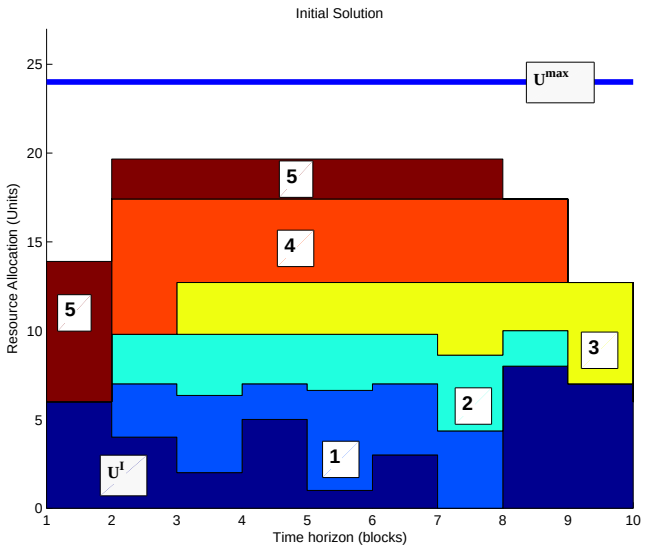


Figure 5-16 – Initial solution of *Guillotine Cut and Layering*

5.4.1 Greedy occupation heuristic

The objective of the layering process is to flatten the resource peak. Hence it leaves a lot of available resources. The greedy occupation heuristic tries to make the most of the resource left to attribute to jobs, in other to accelerate the completion times. The greedy occupation process in a nutshell: the heuristic scans every time interval, if there is an

available amount of unallocated resource then the heuristic will distribute greedily to jobs. The Greedy Occupation is based on LNS principle. While there is unconsumed resource left at time k, the tasks with least demand $\tilde{x}_{i^*}$ and starts to process at k will be removed from power-strip. Task i^* will be repackaged greedily into the strip by algorithm 7. The construction of job i^* by algorithm 7 always leads to a feasible solution, since the worst case is already the previous task's resource allocation before removal.

Algorithm 7: Greedy resource consumption

Data: Task i and starting interval k
Result: New resource allocation u_i
while $\tilde{x}_i > 0$ **do**
  $u_{i,k} := \min\{U_k - U - \Omega[k], \bar{u}_i\}$;
  $u_{i,k} := \max\{u_{i,k}, \underline{u}_i\}$;
  $\tilde{x}_i := \tilde{x}_i - u_{i,k}$;
  $k := k + 1$;
end

Algorithm 8: Greedy Occupation

Data: Actual resource allocation matrix u
Result: New resource allocation u
initialization;
$Accu_i = 0 \quad \forall i \in [n]$;
foreach $k \in \{0..H\}$ **do**
  foreach *job* $i \in \{1..n\}$ **do**
    if $u_{i,k} > 0$ *and* $u_{i,k-1} = 0$ `// If task` i `starts at` k
    then
      Add i to $\mathcal{Q}$
    end
  end
  $sorted\mathcal{Q}$ = sort $\mathcal{Q}$ by $\tilde{x}$;
  $\bar{U} = U_k - U_\Omega[k]$;
  while $sorted\mathcal{Q} \neq \emptyset$ **and** $\bar{U} > 0$ **do**
    Exact i from $sorted\mathcal{Q}$;
    Remove i from Ω;
    $\bar{U} = \bar{U} - u_{i,k}$;
    Allocate resource to i by algorithm 7;
    $\bar{U} = \bar{U} + u_{i,k}$;
  end
end

Remark At each iteration of k, the algorithm has to sort the queued involved jobs, algorithm 8 can be computed in time average $O(H.nlogn)$

5.4.2 Shuffle heuristics

We introduce two more LNS methods to improve the initial solution named ***Free-form shuffle*** and ***Rectangular-form shuffle***. Those two heuristics are classified as "shuffle" heuristic because of their methods of solution destruction. The idea of a shuffle procedure comes from the notation of the k-exchange of very large scale neighbour search [3]. In our heuristic, the k-exchange is implemented as r-random destructions and reconstitutions. Different than the Greedy Occupation which destroys only one job at a time, the shuffle heuristics aim to a more radical approach. At each iteration, the shuffle process decides to destroy k randomly chosen jobs in the power strip. Then, the heuristic repacks those jobs into the trip by two ways: greedy resource consumption or moldable resource consumption. To avoid that the heuristic being over-parameterized, we fixed the k-exchange to be 2-exchanges. The heuristic will destroy and reconstitute jobs pair by pair.

Free-form shuffle As an input, given a number of randoms pairs of jobs, noted $nPairs$. For each pair, jobs will be removed from the power strip and then it tries to re-enter the power strip by the Greedy resource consumption (algorithm 7). The algorithm is detailed in 9. Any better feasible solution found during the shake becomes new incumbent.

Algorithm 9: Free form shuffle algorithm

Data: Actual resource allocation matrix u, maximum number of randoms pairs $nPairs$
Result: New resource allocation u
initialization;
$Paired_{i,j} = false \quad \forall i, j \in [n]$;
for $pair = 1$ *to* $nPairs$ **do**
  Creating randomly jobs i and j;
  foreach *job* z *in* $\{i, j\}$ **do**
    $startTime = r_z$;
    $feasible =$ false ;
    while *not* $feasible$ **do**
      Greedily repack job z using algorithm 7;
      if *new solution found* **then** $feasible =$ true;
      else $startTime = startTime + 1$;
    end
  end
  if *Better solution found* **then** Incumbent = New solution;
end

Remark Let the number of pairs $nPairs = cn; c > 0$. The greedy positioning can be implemented in time $O(H)$. Thus, algorithm 9 can be implemented in time $O(nH)$.

Rectangular-form shuffle The Retangular-form shuffle has the same destruction mechanism with the free-form shuffle algorithm. However, the reconstruction of tasks form of moldable resource allocation, i.e. tasks have to consume constant resource. That comes the name of the method, because the packing item is considered to be rectangle.

Algorithm 10: Rectangular-form shuffle algorithm

Data: Actual resource allocation matrix u, maximum number of randoms pairs $nPairs$
Result: New resource allocation u
initialization;
$Paired_{i,j} = false \quad \forall i, j \in [n]$;
for $pair = 1$ *to* $nPairs$ **do**
  Creating randomly jobs i and j;
  foreach *job* z *in* $\{i, j\}$ **do**
    $p_{max} = min\{d'_z - r'_z, \frac{x^*_z}{u^{min}_z \Delta t}\}$;
    $p_{min} = \frac{x^*_z}{u^{max}_z}$;
    $p_i = p_{max}$;
    $feasible = false$;
    repeat
      Constructing shiftable interval $J_i = \langle r_i, d_i, p_i \rangle$;
      for $\phi_i = 0..\phi_{max}$ **do**
        if $\exists \mathcal{P}_{\phi_i}$ **then**
          $feasible = true$;
          $\phi^* = \phi_i$;
          $p^*_i = p_i$;
          Exit For ;
        end
      end
      $p_i = p_i - 1$;
    until $feasible = true$;
  end
  if *Better solution found* **then** Incumbent = New solution;
end

Remark Let the number of pairs $nPairs = cn; c > 0$. The positioning of each shiftable interval can be implemented in time $O(H^2)$. Thus, algorithm 10 can be implemented in time $O(nH^2)$.

Table 5.3 – Summary of three LNS methods

Heuristics	**Destructions**	**Constructions**
Greedy Ocupation (A)	Least demand removal	Greedy resource allocation: task using resource as much as possible to complete as soon as possible
Free-form shuffle (G)	k-random selection	
Rectangular-form shuffle (R)		Greedy moldable ressource allocation: job having constant amount of resource during its whole processing times

Since all the components have been introduced, we would insist on the basic differences between two versions of the HLA heuristic we will test on the numerical experimentation section: the HLA-RAG and HLA-ARG. In the former version of the heuristic, HLA-RAG, the solution is polished in the following order: Rectangular-form shuffle, Greedy occupation, and Free-form shuffle. In the latter version, HLA-ARG, this order is changed to: Greedy Occupation first, then Rectangular-form shuffle and Free-form shuffle. By the sake of simplicity, in the illustrative example, we take into account only the HLA-RAG version.

5.4.3 Illustrative example

Let us continue to illustrate with the previous described example in the last section. After having the initial solution (see figure 5-16), jobs' resource allocation will be re-assigned in a rectangular form and also aiming to finish as soon as possible by *Rectangular-form shuffle*.

In order to make the most of the resource left to finish all the jobs sooner, the last solution shown in figure 5-17 will be introduced in *Greedy Occupation process* which result is shown in figure 5-18. According to this result, 2 more units of times are minimized for job 4 in the objective function.

Finally, aiming to find any other jobs' combinations should lead to better solution, a *Greedy shuffle in free-form* is carried out. Comparing to the best solution found so far (Fig. 5-18), this new solution (Fig. 5-19) having job 2 finished -4 intervals earlier while having job 4 finished 2 intervals later. In brief, 2 unit of times is minimized in total for the

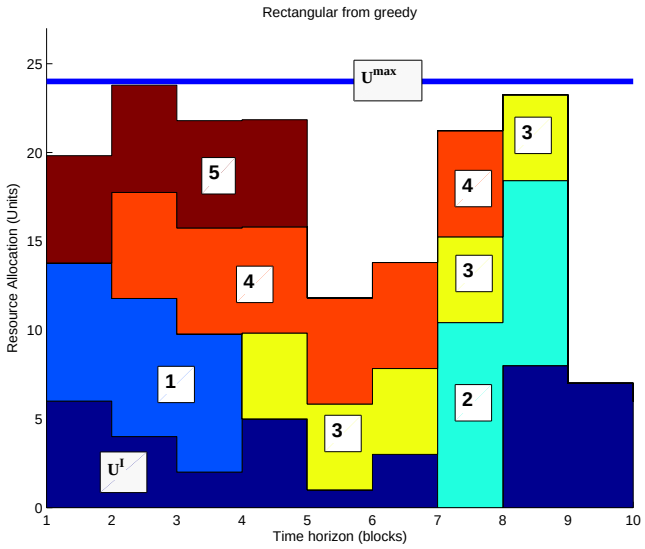


Figure 5-17 – Solution after *Rectangular-form shuffle* procedure

objective function.

5.5 Computational study

Computational experiments are carried out to evaluate the results obtained by heuristics HLA with the ones obtained from solver CPLEX. All the algorithms and test instances generators were implemented in Matlab 2009, the MILP model was implemented in IBM ILOG CPLEX Optimization Studio 12.6. Testing computer used CPU Intel core i5 3.20GHz with 8GB RAM.

Heuristic parameters tuning There are two principal parameters for the *HLA* heuristic: the time-out (noted $timeOut$) for layering process and the number of shuffle-pair for the greedy shuffle (noted $nPairs$). Due to our industrial requirement introduced in our study case, for a parking of 30 EV to be scheduled in a horizon of 12 hours, discretized to 5 minutes per decision interval, $12.60/5 = 144 \simeq 150$, we have to find a feasible solution within 20 seconds. Unless, the parking will be operated in fail-safe mode. Since the layering and cut should be calculated in time $O(nH^2)$, we fix the $timeOut = \frac{nH^2}{33750}$. For $n = 30, H = 150$ then $timeOut = 20$ seconds. the $nPairs$ is fixed to n^2 to assure the diversity of the greedy

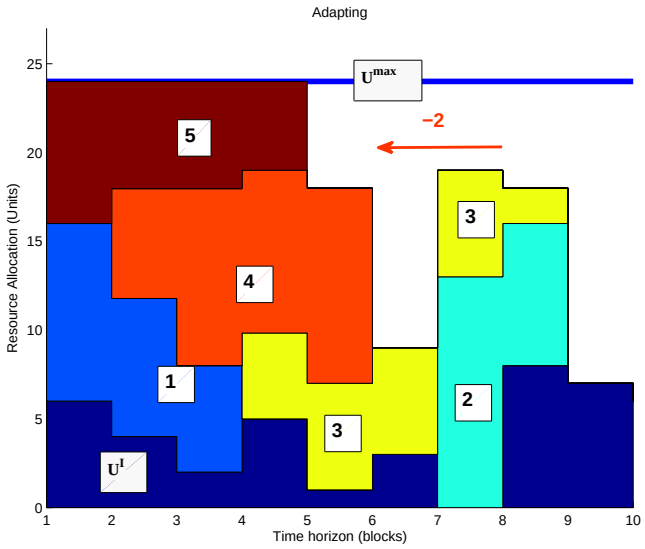


Figure 5-18 – Solution after *Greedy Occupation* procedure

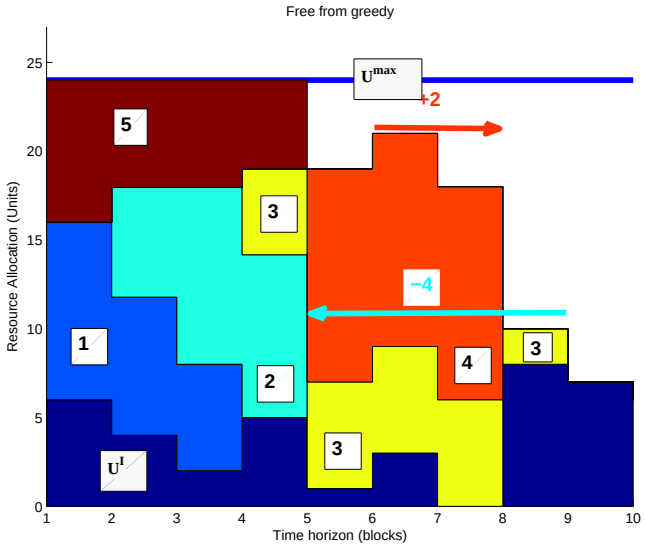


Figure 5-19 – Solution after *free-form shuffle* procedure

shuffle procedure.

Experimentation setting There are generally two groups of instances to be tested in this section. The first group is called small instances group where $n \in \{10, 20\}$ and $H \in \{40, 60, 80\}$. The other one is called large instances group where $n \in \{50, 100, 150, 200\}$

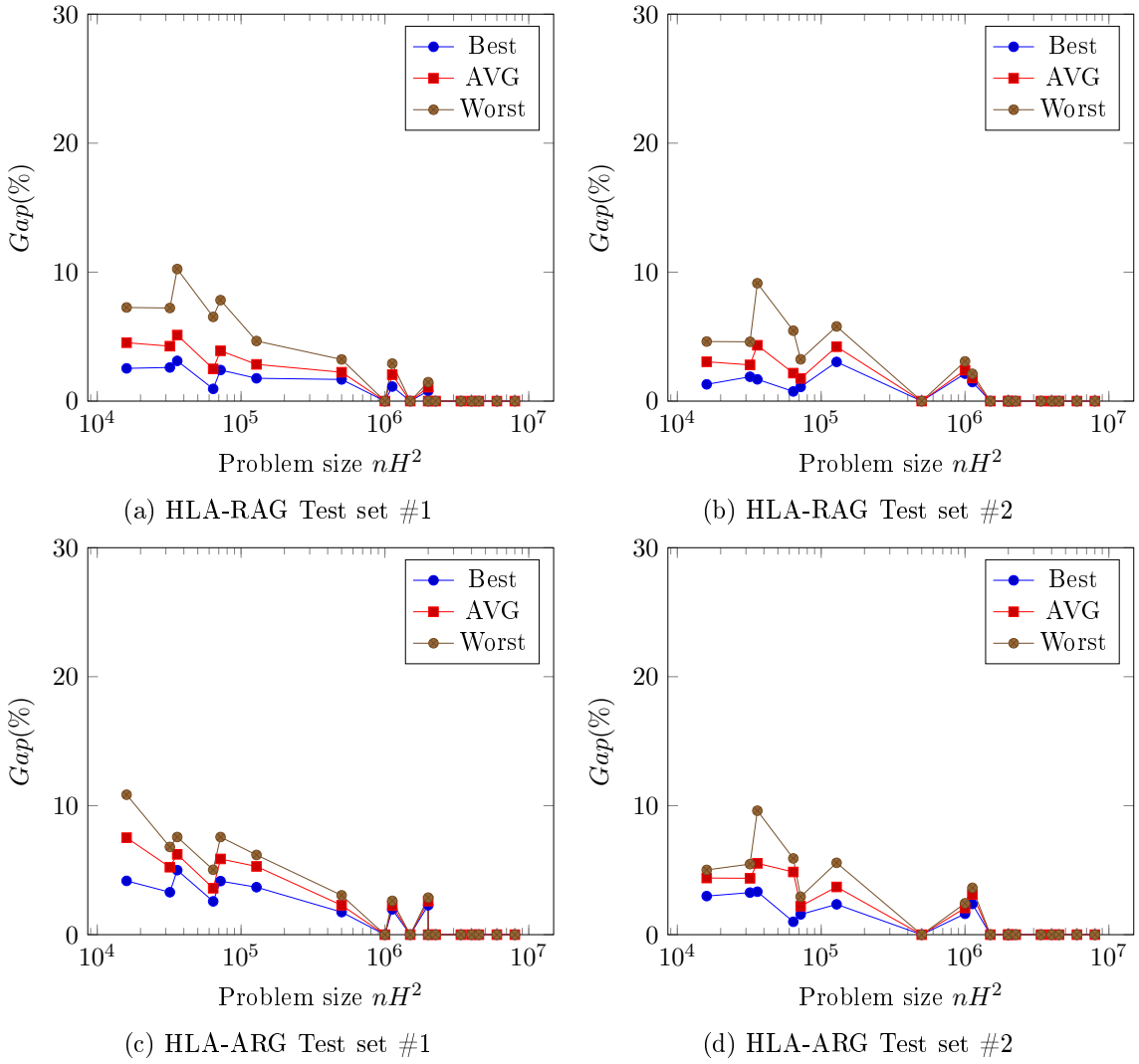
and $H \in \{100, 150, 200\}$[1]. For each pair of n and H, we generate two random instances. The *HLA-ARG* and *HLA-RAG* repeatedly solve each instance 100 times to track the best objective value and execution time found, also the average and worst one. This statistical results help us to observe and understand the behavior of HLA heuristic since its components are based on randomness. The test protocol was the same as the one described in section 3.5.1.

As far as we know, there are no algorithms to cope with the considered problem. The variation of the total available resource causes the decision problem to find whether it is feasible or not to be NP-Hard. For that reason, it is almost impossible to adapt another heuristics on the literature to solve our problem. Hence, we introduce simply the greedy algorithm based on shortest-processing-time (SPT) rule and earliest-due-date rule (EDD) [49]. Finally, the CPLEX solving time-out is set to 1800 seconds.

We introduce in the following list the notations used in the result tables of the computational tests

- $ObjVal$ the objective value of solution found by the tested methods. For the heuristic, there are also the $Best$ - the best objective value, AVG and $Worst$ the average and the worst value of 100 solving attempts.
- $CPUTime$ the execution times measured in second of each method. HLA heuristic also have $Best$, AVG and $Worst$ values.
- Gap is the gap (measured in percentage) between the objective found by heuristics and CPLEX. $Gap = \frac{ObjVal^{Heuristic} - ObjVal^{CPLEX}}{\max\{ObjVal^{Heuristic}; ObjVal^{CPLEX}\}}$
- $noInt$ denotes a status where CPLEX can not find an integral for the input instance, yet the input instance is proven to be feasible. In this case, the $ObjVal^{CPLEX}$ corresponding to this instance takes value infinity. If the heuristic can find a feasible solution in this case then $Gap = 0\%$.
- $noSol$ indicates that the heuristic(s) can not find a feasible solution for the given instance. In this case, the $Gap = 100\%$.

[1] All the input data and solutions found by solver CPLEX and *HLA-RAG*, *HLA-ARG* can be found at the link: http://losi.utt.fr/_resources/documents/TestInstance_Results_NNQ_Paper2015.zip



(a) HLA-RAG Test set #1

(b) HLA-RAG Test set #2

(c) HLA-ARG Test set #1

(d) HLA-ARG Test set #2

Figure 5-20 – *Gap* found by HLA throughout different problem sizes nH^2. The horizontal axes are in logarithmic scale.

Results on objective values found by heuristic (HLA-RAG, HLA-ARG, EDD and SPT) are shown in table 5.4. For ease of reading, best values found by heuristic(s) are in bold, objective value found by CPLEX are in italic. The rates of finding feasible solutions of EDD and SPT are small, accordingly 33% for EDD and 10% for SPT. This failure in finding feasible solutions can be explained by the NP-Hard nature of the feasibility test. CPLEX successfully finds integral solutions (with time limited to 1800s) in 57% of the tested instances, in which 15% of the solutions found are optimal. By contrary, HLA heuristics yield a 100% rate of finding feasible solution.

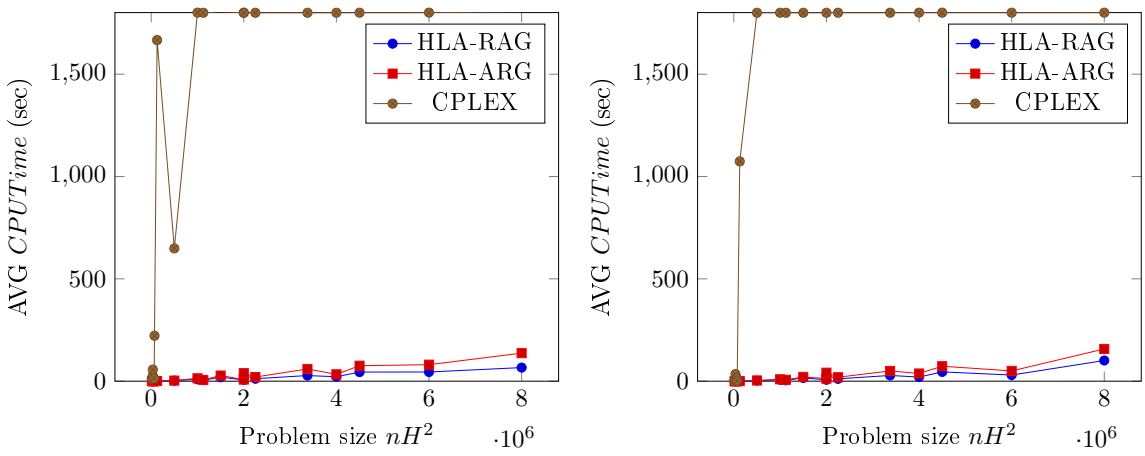
Figure 5-20 plots the evaluation of the best, the average and the worst *Gap* found by

Table 5.4 – Objective values of solutions found by heuristics: HLA-RAG, HLA-ARG, EDD, SPT and by solver CPLEX.

n	H	#	HLA - RAG			HLA - ARG			CPLEX	EDD	SPT
			Best	**Average**	**Worst**	**Best**	**Average**	**Worst**			
10	40	1	**236**	**240.91**	**248**	240	248.7	258	*230*	noSol	noSol
		2	**230**	**234.14**	**238**	234	237.42	239	*227*	245	noSol
	60	1	**353**	**360.44**	381	360	364.72	370	*342*	noSol	noSol
		2	**354**	**363.74**	**383**	360	368.39	385	*348*	420	noSol
	80	1	**419**	**425.62**	444	426	430.49	437	*415*	472	499
		2	**401**	**406.84**	**421**	402	418.32	423	*398*	440	440
20	40	1	**422**	**429.27**	443	425	433.7	**441**	*411*	455	noSol
		2	**423**	**426.98**	**435**	429	433.97	439	*415*	443	noSol
	60	1	**663**	**673.26**	702	675	687.32	**700**	*647*	755	noSol
		2	**635**	**639.28**	649	638	642.22	**647**	*628*	noSol	noSol
	80	1	**958**	**968.6**	**987**	977	993.55	1003	*941*	noSol	noSol
		2	856	866.55	881	**850**	**861.9**	**879**	*830*	903	953
50	100	1	**2684**	**2699.5**	2727	2686	2700.7	**2722**	*2639*	noSol	noSol
		2	2933	2952.7	2973	**2931**	**2945.7**	**2959**	*noInt*	noSol	noSol
	150	1	**4353**	**4394.3**	4433	4390	4403	**4420**	*4304*	noSol	noSol
		2	**4123**	**4135.9**	**4150**	4161	4192.7	4215	*4062*	4348	noSol
	200	1	**4916**	**4930.4**	**4950**	4991	5007.3	5021	*4877*	5146	noSol
		2	**5553**	**5581.8**	5626	5550	5591.6	**5619**	*noInt*	noSol	noSol
100	100	1	**6209**	**6242.3**	**6271**	6253	6312	6390	*noInt*	noSol	noSol
		2	**5819**	5835.2	5876	5789	**5815.6**	**5837**	*5695*	noSol	noSol
	150	1	**8466**	**8491.1**	8528	8486	8504.4	**8524**	*noInt*	noSol	noSol
		2	**8127**	**8156**	**8182**	8220	8246.4	8265	*noInt*	noSol	noSol
	200	1	12129	12161	12238	**12106**	**12155**	**12193**	*noInt*	noSol	noSol
		2	**11748**	**11794**	**11840**	11838	11875	11913	*noInt*	noSol	noSol
150	100	1	8929	8969.1	9014	**8928**	**8957.9**	**8998**	*noInt*	noSol	noSol
		2	**8371**	**8419.3**	8485	8412	8441.3	**8484**	*noInt*	noSol	noSol
	150	1	**12792**	**12850**	**12905**	12834	12919	12966	*noInt*	noSol	noSol
		2	**13387**	**13427**	**13469**	13449	13496	13561	*noInt*	noSol	noSol
	200	1	17629	17694	17762	**17614**	**17687**	**17743**	*noInt*	noSol	noSol
		2	16486	**16505**	**16531**	**16446**	16530	16581	*noInt*	noSol	noSol
200	100	1	**11553**	**11571**	**11598**	11586	11629	11659	*noInt*	noSol	noSol
		2	11618	11653	11669	**11516**	**11554**	**11573**	*noInt*	noSol	noSol
	150	1	**15756**	**15791**	**15832**	15842	15887	15946	*noInt*	noSol	noSol
		2	**17678**	**17701**	**17746**	17762	17799	17845	*noInt*	noSol	noSol
	200	1	21870	21893	21936	**21755**	**21837**	**21884**	*noInt*	noSol	noSol
		2	**23935**	**24007**	**24045**	24016	24128	24287	*noInt*	noSol	noSol

Table 5.5 – *CPUTime* of heuristics: HLA-RAG, HLA-ARG, EDD, SPT and of solver CPLEX.

n	H	#	HLA - RAG			HLA - ARG			CPLEX	EDD	SPT
			Best	Average	Worst	Best	Average	Worst			
10	40	1	0.08	**0.08**	0.15	**0.07**	**0.08**	**0.08**	*11.81*	noSol	noSol
		2	0.07	0.07	0.08	0.07	0.07	0.07	*1.2*	**0.002**	noSol
	60	1	**0.13**	**0.15**	**0.17**	0.14	**0.15**	**0.17**	*56.77*	noSol	noSol
		2	**0.14**	**0.16**	0.19	0.16	0.17	**0.17**	*34.5*	0.003	noSol
	80	1	0.21	0.22	0.25	0.24	0.25	0.36	*13.47*	**0.003**	**0.003**
		2	0.15	0.16	0.19	0.15	0.16	0.19	*14.73*	**0.004**	**0.004**
20	40	1	0.19	0.20	0.22	0.17	0.18	0.21	*25.75*	**0.004**	noSol
		2	0.19	0.20	0.21	0.18	0.19	0.20	*10.16*	**0.007**	noSol
	60	1	0.34	0.36	0.41	0.37	0.40	0.42	*221.98*	**0.007**	noSol
		2	0.27	0.28	0.29	**0.25**	**0.26**	**0.28**	*2.13*	noSol	noSol
	80	1	**0.59**	**0.66**	**0.73**	0.79	0.81	0.91	*1666.7*	noSol	noSol
		2	0.53	0.56	0.64	0.56	0.60	0.84	*1074*	0.009	**0.008**
50	100	1	**2.62**	**2.66**	**2.73**	2.68	2.84	2.97	*648.83*	noSol	noSol
		2	**2.63**	**2.75**	**2.93**	3.11	3.45	3.85	*1800.1*	noSol	noSol
	150	1	**4.63**	**5.01**	**5.38**	**5.37**	6.15	6.92	*1800.1*	noSol	noSol
		2	4.23	4.44	4.70	6.46	6.87	7.07	*1800.2*	**0.007**	noSol
	200	1	6.55	6.65	6.85	7.48	7.78	8.18	*1800.1*	**0.01**	noSol
		2	**6.75**	**7.05**	**7.44**	11.26	12.48	13.12	*1800.2*	noSol	noSol
100	100	1	**8.77**	**9.24**	**9.98**	12.55	13.93	15.37	*1800.8*	noSol	noSol
		2	**6.44**	**6.92**	**7.26**	8.31	9.11	9.97	*1800.2*	noSol	noSol
	150	1	**11.46**	**12.06**	**12.72**	18.21	20.31	22.15	*1800.2*	noSol	noSol
		2	**11.08**	**11.52**	**11.94**	17.18	19.27	21.03	*1800.3*	noSol	noSol
	200	1	**19.82**	**21.65**	**24.58**	28.51	33.53	39.65	*1800.3*	noSol	noSol
		2	**18.85**	**20.10**	**21.12**	35.01	37.54	40.18	*1800.6*	noSol	noSol
150	100	1	**17.73**	**19.17**	**21.08**	25.40	27.54	29.67	*1804.7*	noSol	noSol
		2	**14.83**	**15.64**	**17.07**	19.24	20.86	21.73	*1800.2*	noSol	noSol
	150	1	**26.17**	**27.57**	**28.61**	55.69	59.62	61.49	*1800.3*	noSol	noSol
		2	**26.20**	**28.00**	**30.22**	41.90	50.08	57.03	*1800.4*	noSol	noSol
	200	1	**41.58**	**44.91**	**49.56**	72.66	80.71	86.18	*1800.7*	noSol	noSol
		2	**29.29**	**29.99**	**30.91**	43.12	50.12	55.09	*1800.5*	noSol	noSol
200	100	1	**27.18**	**27.91**	**29.83**	35.56	39.12	43.25	*1800.3*	noSol	noSol
		2	**27.50**	**28.76**	**30.16**	40.08	41.46	43.36	*1801.2*	noSol	noSol
	150	1	**42.35**	**44.49**	**47.26**	70.75	75.60	80.15	*1800.5*	noSol	noSol
		2	**43.36**	**45.35**	**49.04**	65.69	73.15	80.70	*1800.6*	noSol	noSol
	200	1	**64.74**	**66.45**	**68.12**	123.48	136.93	151.12	*1804.5*	noSol	noSol
		2	**93.22**	**101.01**	**108.15**	146.89	157.50	169.92	*1800.7*	noSol	noSol



(a) Average $CPUTime$ Test set #1

(b) Average $CPUTime$ Test set #2

Figure 5-21 – Average $CPUTime$ throughout different problem sizes nH^2

two HLA: HLA-RAG and HLA-ARG throughout the tested instances according to its size nH^2. The *Gap* of EDD and SPT are not displayed due to two reasons: the simplicity of presentation and the small rate of feasibility of EDD and SPT. With data in table 5.4 one can calculate the *Gap* of EDD and SPT which generally varies between 10% to 18%. The *Gap* found by both of the HLA heuristics decrease over the problem size in terms of nH^2. By the average *Gap* found, one can conclude that both HLA heuristic perform well with the two sets of tests. In detail, one can find that the HLA-RAG shows slightly better solutions in the best and average cases while HLA-ARG maintains better the worst *Gap*. HLA-RAG also keeps a stricter margin between the best case and the worst case, hence it is more stable in terms of solution qualification.

The execution times resulted from each test are shown in table 5.5. For the ease of reading, best solving times of heuristic(s) are in bold, $CPUTime$ of CPLEX are in italic. Thanks to the simplicity of implementation, EDD and SPT yields the best results in terms of execution time when it can find a feasible solution. The CPLEX works at its best in small instances where $n \leq 20$ and $H \leq 80$. The $CPUTime$ of CPLEX escalates quickly on medium and large instances yet it still hard to find a feasible solution. The execution time of both HLA heuristics displayed in figure 5-21 is proportional with nH^2 which corresponds to the theoretical complexity calculation. In the largest instances, $n = 200$ and $H = 200$, HLA-RAG takes less than two minutes to find final solution while HLA-ARG takes less than 2.7 minutes. The execution times of both the heuristics to find final solutions is quick

compared to the instance size to deal. The gap between the best and the worst time found for each instance is relatively small, thus both of the two HLA versions are stable in terms of execution time. For more detail, HLA-RAG outperforms HLA-ARG by having in average shorter execution times for both of the tested sets. For small instances, the difference between the best and the worst *Gap* is relatively big but the *CPUTimes* is really small. If the real-life implementation of the heuristic allows a larger execution time, one can profit from the time left to repeat HLA for more than one run, hence the probability of finding the best *Gap* solution is high. For instance, in our industrial implementation, the time limit is 20 seconds than we can have around 100 runs for each input instance with real-life input $n = 25$ $H = 48$.

In the actual industrial case, the heuristic has proven a reasonably good performance. Specifically, according to the implementation tests of *HLA-RAG* dealing with our latest industrial problems containing up to 25 jobs in a 12-hours-scheduling scheme, each interval lasts 15 minutes (48 intervals in total), the average derivation found between the solutions obtained by *HLA-RAG* and optimal solution resulted from CPLEX is 1.7% while the execution time-out of the heuristic is set to 20 seconds.

5.6 Conclusion

In this chapter, we introduced a family of constructive heuristics to deal with the ACPV problem. The strip packing problems inspires the heuristics. Based on the Large Neighbour Search principles, we have developed one solution construction method with three solution improvements methods. During the statistical test, the solution construction phase proved very efficient to find the first feasible solution. Also, the strip packing approach and constructive way of establishing the solution makes this heuristic extremely flexible: One can add or remove the time-window constraints, add more restrictions to the resource consumption to the heuristic. We also introduced three LNS methods: Greedy Occupation, Free-form shuffle and Rectangular form shuffle. We conducted numerical tests to compare the two versions of the heuristic, distinguished by the orders of the methods applied to the solution improvement phase. Those two versions work well with the proposed problem, proving an adamant computational advantage in terms of execution time. The HLA-RAG

heuristic yielded the best results over most of the tested instances.

We have now efficient deterministic scheduling algorithms. In the next chapter, we present how a deterministic algorithm can be apply to cope with the uncertainties of real-life scheduling problems. We use a predictive-reactive scheduling framework to cope with the forecasting problem and the online scheduling.

The work done in this chapter is subject to one publication [56].

Chapter 6

Simulation based forecast and online scheduling

Contents

6.1 Introduction 130

6.2 Preliminaries 131

6.2.1 Terminologies 131

6.2.2 Rescheduling methodology 132

6.3 Method outline 133

6.3.1 Predictive-reactive rescheduling 133

6.3.2 The power and cost trade-off 134

6.4 The simulation based forecast 136

6.4.1 Fixed time-horizon simulator 137

6.4.2 Fixed power bandwidth simulator 138

6.4.3 Case study 139

6.5 Online scheduling 143

6.5.1 From deterministic scheduling to online scheduling 143

6.5.2 The decision making policies 143

6.5.3 Reactive rescheduling 144

6.6 Conclusion 146

6.1 Introduction

After analysing, formulating and developing solution procedure for the problem, one would raise a question: How to implement all of the latter to the industrial application? For that reason, we introduce in this chapter a predictive-reactive rescheduling method to incorporate the algorithmic research to the technical implementation of a forecast model and an online scheduling.

As we know in the first chapter, a corporate electric vehicle charging management tends to minimise the cost and maximise the service quality. To do that, we identify three decision levels that should be made.

At installation: How much total power does one have to subscribe from the electrical supplier? This decision is a trade-off between the fixed cost of power subscription and the service level. The more bandwidth is allocated monthly, the more costly the annual or monthly subscription cost. For instance, the EDF supplier in France costs around 500 euros for the annual subscription of a power 24 kVA compared to 700 euros per year for 36 kVA. [22].

At daily operation: On which schedule baseline can the on-line management depend on for the day-to-day operation? With a fixed power bandwidth, one has to define a good schedule start time to assure the service level. In France, overnight charging scheme usually starts at 10 PM and ends at 6 AM to profit the off-peak low electric cost. In some days, due to heavy charging demands, one has to decide to start the procedure earlier.

Online scheduling: Facing with certain controllable activity (charging takes longer than predicted, charging power is lower than provided power...) or uncertainties (car arriving late in the parking, unplugging before end of charge...), how can the management central react and update a new schedule?

Hence, we construct a rescheduling method based on the predictive-reactive manufacturing to deal with the three decision levels that we may note: long term, short term and operational.

This chapter will be segregated in 4 parts. First, we introduce the problem and the need for an aid decision tool. Preliminaries on the remanufacturing help us to introduce some basic conceptions on section 6.2. We start with the study case and a simple version of the HLA to cope with the ACPF problem (moldable charging task). Then, in section 6.4 we introduce the simulation based forecast, which would respond to the two first concerns in the long term and short term decisions. Section 6.5 outlines the online scheduling method of the problem. At the end of the chapter, we draw conclusions on the simulation based forecast and the online scheduler.

6.2 Preliminaries

Since the 90s, many studies have discussed many fragments of the idea of a predictive-reactive remanufacturing. However, there exist no official or standard definition and classification. According to our knowledge Vieira et al. [76] were the first to address the conception of a re-manufacturing method systematically. Later, Van de Vonder et al. [70] discussed in more details the project scheduling procedure under the optic of the predictive-reactive rescheduling. First, we would mention some definitions and notations concerning the method.

6.2.1 Terminologies

Rescheduling: The process of adapting or updating the actual production schedule in order to respond or adapt to changes or disruption in the real time event.

Rescheduling environment: the characteristic of the set of jobs should be scheduled.

Rescheduling strategy: Decision that one should regenerate the schedule or not.

Rescheduling policy (method): the description of how and when the schedule is regenerated.

Disturbance: An unexpected event that does not correspond to the actual schedule.

Baseline schedule: Any feasible solution of the deterministic schedule problem input.

Decision maker: A person, or a stand-alone scheduler, which decision what is the concrete reaction/respond for each action/event caught by the system. This decision can be an reallocation of resource, a start or a stop of a task execution.

6.2.2 Rescheduling methodology

We would cite some definitions and classifications done by Vieira et al. [76] to introduce the conception of the rescheduling framework. As we have mentioned in the previous part of the section, a scheduling framework includes an environment, strategies and policies. Concerning the rescheduling environment, if the set of jobs is finite then it is called static. Otherwise, it is called dynamic. In the case of static rescheduling environment, if all the information about jobs to schedule is given then we make the plan in the deterministic environment. If we have to cope with some level of uncertainties, then the scheduling environment is then stochastic.

Regarding the rescheduling strategies, if there a no schedule baseline, hence there are no updating of the baseline, the immediate task is dynamically put in the system. In that case, the rescheduling strategy is called dynamic, which uses dispatching rules or control-theoretic. The dynamic rescheduling strategy is comparable to the implicit optimisation approach that we introduced in chapter 2 because it does not create production schedules. Otherwise, if the rescheduling strategy is made up of a baseline and a procedure of baseline updating, it is then called predictive-reactive. The updating of the baseline schedule can be periodic or event-driven. A periodic policy updates the schedule baseline at a regular interval. Hence this policy is not suitable for online scheduling. For that reason, one also considers the event-driven policy: the schedule baseline will be changed at every new event that had not been taken into account.

The rescheduling consists of two main methods: schedule generation and schedule repair. To cope with uncertainties, one has two choices: whether we generate a robust schedule, which can be understood as a feasible schedule with adjustment to deal with disruptions; or we reschedule each time we face unexpected events, which is called schedule repair. The schedule reparation can be complete, which change all the allocation of tasks already started and tasks in the waiting queue, or partial, which reschedule only tasks

have not started yet. There also exists the right-shift/left-shift schedule reparation. Figure 6-1 resumes the classification of the rescheduling framework environments, strategies and methods.

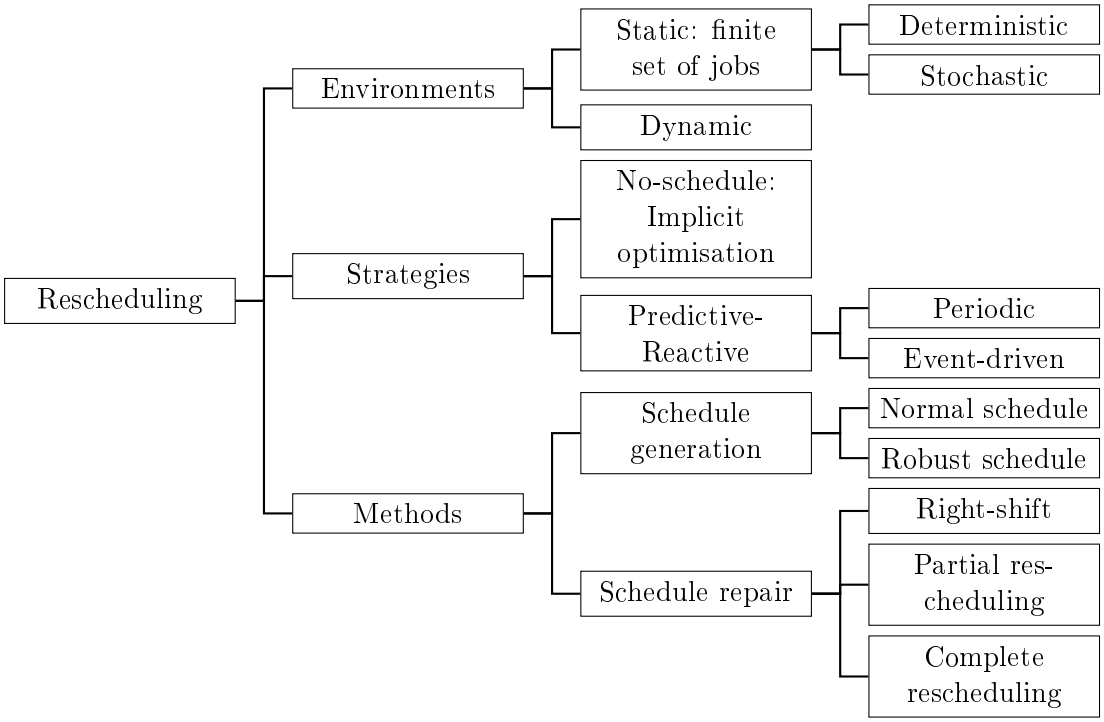


Figure 6-1 – The rescheduling framework classification

6.3 Method outline

6.3.1 Predictive-reactive rescheduling

The overall predictive-reactive rescheduling can be resumed in figure 6-2. Each real life event has a double usage: it enriches the database and triggers the on-line reactive rescheduling module. An optimal power bandwidth is chosen as detailed in the previous section for the long-term schedule baseline. Every day, the database will extract a small number of events (from 14 to 25 events) to estimate the parameter of the customer behaviours random distribution, as well as estimate the parameter of the daily demand. Thence, the scheduling length variation simulator gives us the relation between the probability to have a feasible planning with the earlier shift. An earlier starting time would be made at the

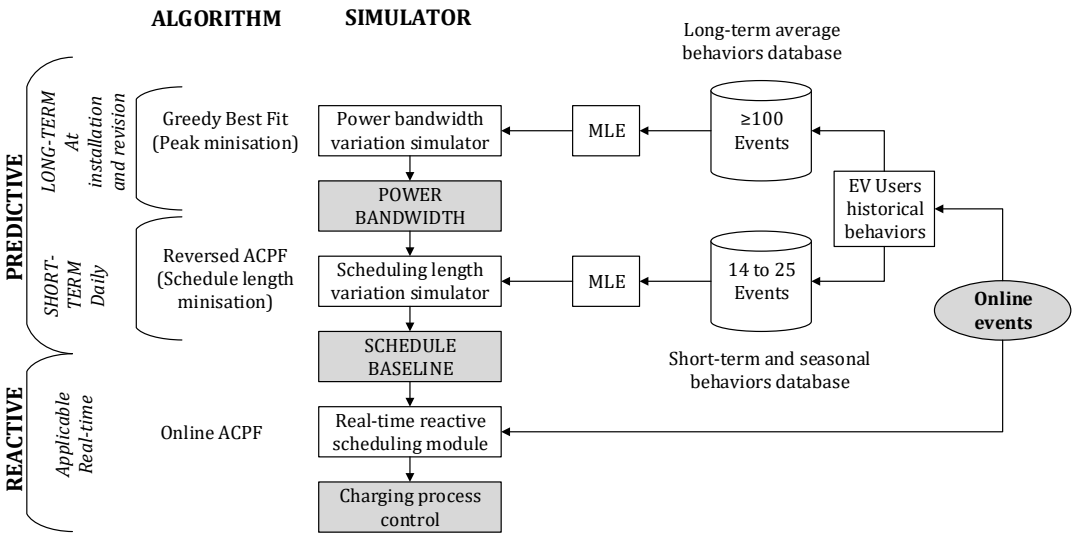


Figure 6-2 – Overall description of Predictive-reactive rescheduling

beginning of the charging process to assure an expected 100% chance of feasibility.

6.3.2 The power and cost trade-off

The industrial problem tackled in this thesis consist of a scheduling problem with a finite set of charging tasks. We know the list of clients in the parking, and all of their past behaviours stored in the database. Every day, each customer comes home and plugs his/her EV to the charge point at random hours. The daily charging energy demand is also random, and depends on the distance travelled that day. However, the expected departure of the car is made by the client throughout the platform web. Hence the stochastic elements of the schedule consisting of the arrival (plugging times) and the daily energy demand of each EV. The rescheduling environment is then static and stochastic.

Regarding the scheduling strategy, we have to make a short-term and a long-term decisions: the power subscribed for the parking and the daily starting time of the scheduling. In figure 6-3, the power subscription is big enough to let the whole charging schedule fit in the off-peak hours. It leads to an expensive fixed cost (power subscription cost) but a minimal variable cost (the electrical usage only in off-peak hours). On the contrary, In figure 6-4, the subscription cost is cut by 2 through the reduction of the power bandwidth.

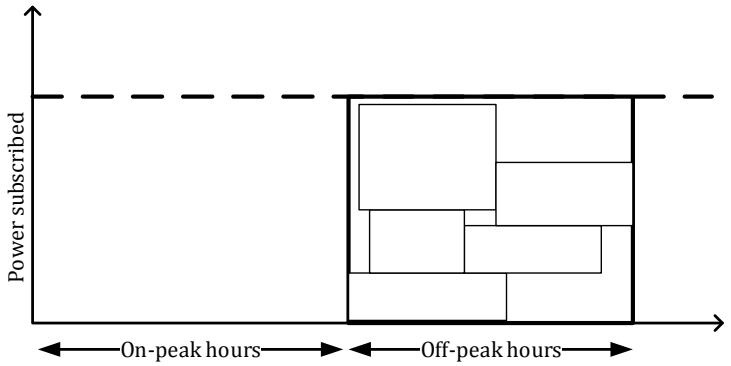


Figure 6-3 – A big power subscribed let the scheduling be completely on the off-peak hours

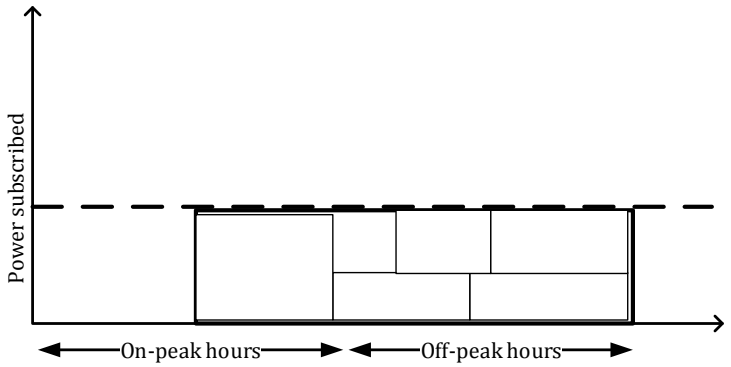


Figure 6-4 – An earlier schedule start (left-shift) for small power subscribed

The variable cost is increased since we have to start the schedule earlier in the on-peak hours pricing plan to assure the feasibility of the planning. For that reason, a good baseline schedule is required. Hence, the scheduling strategy should be reactive-predictive. Normally, a private sector parking contains no more than 30 EV. Given the computational strength of our heuristic, the scheduling problem for a 12h time horizon could be done in less than half a second so it can cope with the on-line event with a reasonably quick response time. Hence, the rescheduling methods should be schedule reparation, driven by new events.

6.4 The simulation based forecast

We present the simulating protocol in two aspects: variation of the power bandwidth and variation of the scheduling length. The statistical results of the two simulators are used to define an optimal power bandwidth and a standard schedule horizon. In addition, during the operation, once the parameters for random generator change according to the new trend of real-life data, schedule baseline can be built to guide the on-line management. For instance, if the baseline states that the current power bandwidth is not enough for all EV to be fully charged in the given time horizon, the whole operation can be started earlier. This framework is resumed in figure 6-5.

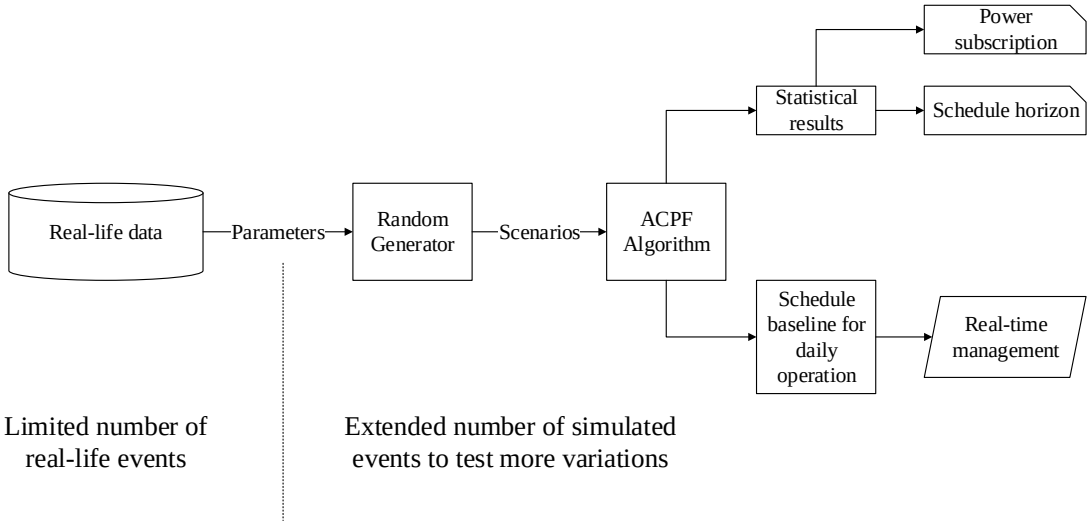


Figure 6-5 – The simulation based forecast

The cost of utilization of the power bandwidth U With the off-peak overnight charging scenario, the using cost of the power bandwidth can be classified onto two costs. The first one is the annual fixed cost of power subscription. The second one is the penalty cost if the charging has to start earlier than off-peak periods. The following equation describes in details those two kinds of costs:

$$z(U) = f_0(U) + P_{BW}(x > U).f_p(U) \tag{6.1}$$

Where:

- $P_{BW}(x > U)$ is the probability the total bandwidth subscribed U cannot satisfy all charging demand within off-peak hours; i.e. the probability of the required power bandwidth to assure feasibility is greater than the subscribed power U.
- $f_0(U)$ is the fixed cost of power subscription U which is usually linear: $f_0(U) = U \times c_0$, c_0 is the subscription cost per KVA.
- $f_p(U)$ is the penalty cost while using total power U due to earlier start before off-peak hours to satisfy all energy demands.

Optimal power subscription should minimize the cost $U^* = \arg\min_u(z(U))$.

6.4.1 Fixed time-horizon simulator

To find the probability $P_{BW}(x > U)$ one should conduct the fixed time-horizon simulator. As being named, this simulator uses the **Greedy Best Fit algorithm** in the assumption that the time-horizon is limited during the off-peak hours. The **GBF** tends to minimize as much as possible the maximum power used for each random instance. All the result of the maximum power used in each instance will be tracked. The statistical result would help us to estimate the distribution of the maximum total bandwidth used. Thus, one can define $P_{BW}(x > U) = 1 - F_{BW}(U)$, where $F_{BW}(U)$ is the probability cumulative function of maximum total power used. The overall procedure is presented in figure 6-6.

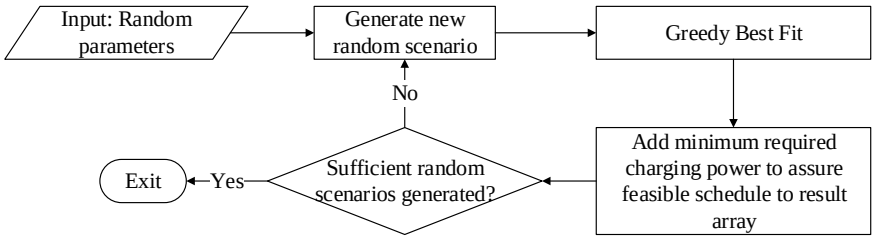


Figure 6-6 – Fixed time-horizon simulator description

6.4.2 Fixed power bandwidth simulator

The reversed GBF algorithm With a fixed power bandwidth, the problem of finding the optimal starting time of the charging procedure is comparable to the problem of makespan optimisation. First, we reverse the time horizon and the time-windows constraints of all the task: the deadline becomes the release date, and the release date becomes the deadline. We execute the greedy best-fit algorithm to find a feasible solution and dichotomise the scheduling horizon. If the optimal schedule length is less than the off-peak hours' duration, then we do not need to left shift the starting time of the charging procedure. Otherwise, the left-shift is equal to the difference between the optimal schedule length and the off-peak duration.

Algorithm 11: Reversed *GBF*

Reverse the time horizon; **foreach** *Job* i **do**
  Swap r_i and d_i
end
while $UB > LB$ **do**
  $IB := \lfloor \frac{UB+LB}{2} \rfloor$;
  Execute *Greedy Best Fit* with time horizon limited to IB ;
  if *Feasible solution found* **then**
    Save new solution;
    $UB := IB$;
  else
    $LB := IB$;
  end
end

Power bandwidth usage cost estimation To estimate the usage cost U of the power subscription, one has to define the penalty function $f_p(U)$. In this paper, we consider the penalty cost as the electrical consumption before the off-peak hours.

$$f_p(U) = U\mathbf{E}_U[\Delta St](c_{peak} - c_{off-peak}) = F_E(U)U\Delta c \tag{6.2}$$

Where $\mathbf{E}_U[\Delta St]$ is the expected duration of the charging operation before the off-peak hours (i.e ΔSt) when the total bandwidth is U. Δc is the difference of price per kW between peak and off-peak hours. For the sake of simplicity, we note $F_E(U) = \mathbf{E}_U[\Delta St]$. Hence the fixed power simulator has a double usage. In strategical level, it can find the value of $f_E(U)$ among varying U to decide the optimal U^* at the parking installation. At daily operational level, if we limit the real-life data into the most recent events, the parameters for random generator would change so the simulator can compute the expected scheduling length for the next charging operation. Thus, one can decide whether the charging operation has to be shifted earlier or not. The overall procedure is shown in figure 6-7.

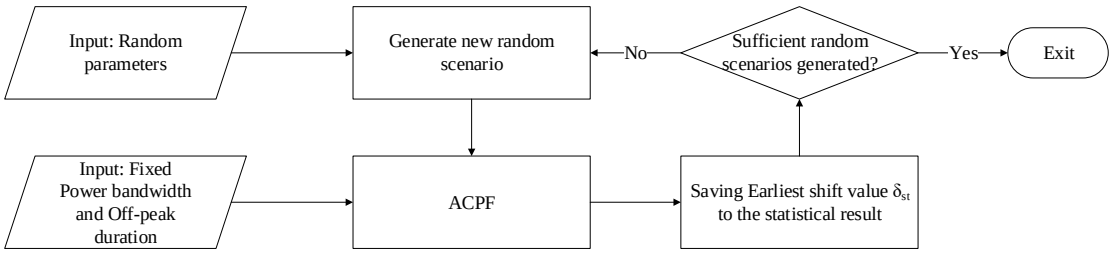
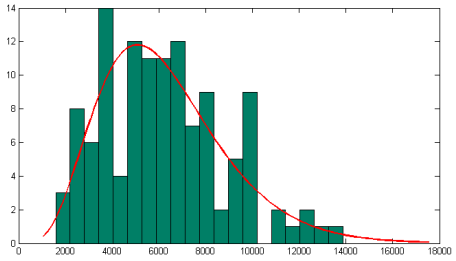


Figure 6-7 – Fixed power bandwidth simulator description

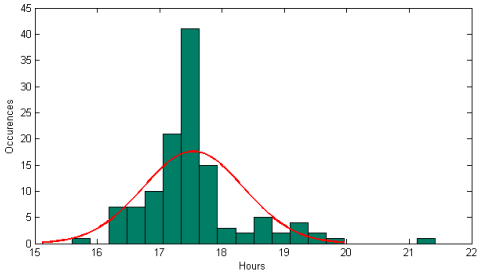
6.4.3 Case study

The case study takes data from our industrial project of the EV charging management for residential parking in France. Historical database of 30 EV is received during the period from Mars 2016 to May 2016. The arrival times are random, but the departure of each vehicle is fixed at 6 AM daily by default. According to the collected data, we choose the gamma distribution to simulate daily consumption and normal distribution to simulate the arrival times. Histograms on real-life data collected (figure 6-8) justify our choices of random distributions for each type of simulated data.

Table 6.1 resumes the customer's random behaviors parameters of 30 EV. For each number of vehicles, 10000 instances will be generated randomly according to the inputted parameters. The off-peak hours is fixed from 22 PM to 6 AM (i.e. overnight charging scheme) to conduct the fixed-time horizon simulator. The **ACPF** algorithm and two simulators are coded in C++. Distribution fit and plotting tools are coded on Matlab 2009b. The result of



(a) Histogram with gamma distribution ($\alpha = 4.5; \beta = 1442$) for the daily consumption over 125 charging events.



(b) Histogram with normal distribution ($\mu = 17.5h; \sigma = 0.8h$) for the daily arrival times over 125 charging events.

Figure 6-8 – Histogram of customer behaviour to justify the choice of random distributions on arrival times and charging demands.

the **fixed-time horizon simulator** is on figures 6-9 and 6-10. Figure 6-9 shows the density of the maximum total charging power occurrences over 10000 random instances. Figure 6-10 states the distribution cumulative function F_{BW} and its distribution fit counterpart. According to the histogram, we can estimate that the total power usage follows a normal distribution whose mean is 16.7 kW (about 14.2% of the sum of all charging power, i.e. **the diversity factor** of electrical use) and its standard deviation is 5.6 kW.

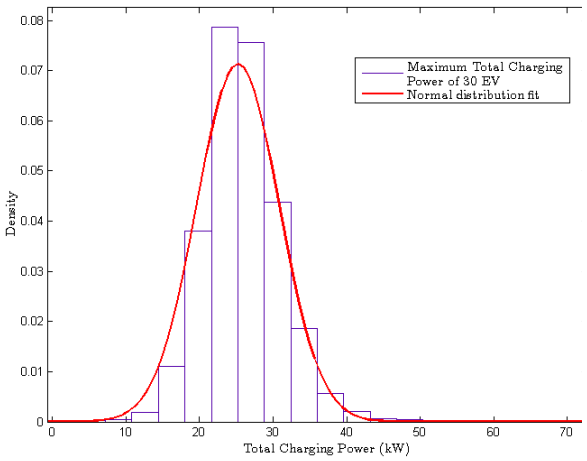


Figure 6-9 – Histogram of the maximum total charging power of 30 EV resulted from 10000 tests

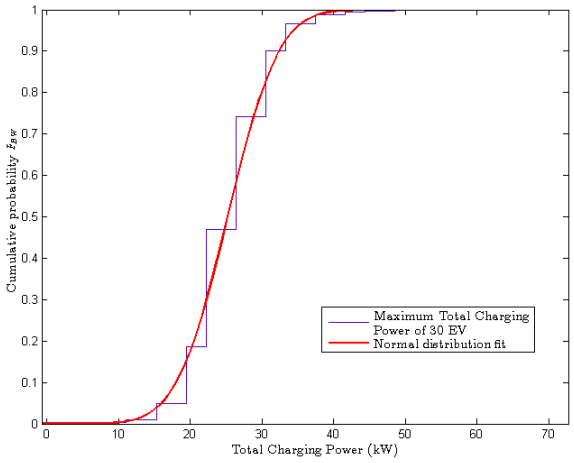


Figure 6-10 – The corresponding probability cumulative function F_{BW} and its distribution fit counterpart

Figure 6-11 shows the result of the **fixed power bandwidth simulator**.

Table 6.1 – Customer random behavior parameters: input for simulators

ID	**Energy demand distribution: Gamma**		**Arrival time distribution: Normal**		**Eff. (%)**	**Power (W)**
	α	β	Mean	Stddv		
1	3.296	1772.90	18.59	1.41	81	7400
2	3.454	1270.39	17.95	1.17	90	7400
3	3.880	1743.37	18.12	1.16	83	3700
4	3.814	1944.83	17.38	0.81	86	7400
5	4.466	986.27	17.56	1.64	90	3700
6	4.192	1319.90	17.62	1.11	93	3700
7	3.440	1203.53	18.03	0.85	82	3700
8	3.224	1016.03	17.61	1.80	94	3700
9	4.225	1438.34	17.91	1.07	91	3700
10	4.084	1015.97	17.07	0.90	87	3700
11	4.396	1947.71	17.59	1.04	86	11100
12	3.663	1441.30	17.45	0.78	86	3700
13	3.457	1527.13	17.22	0.66	86	11100
14	4.136	2023.02	17.85	2.34	88	7400
15	3.668	1829.92	17.78	1.94	88	3700
16	3.855	1523.35	17.71	1.32	92	3700
17	3.322	1294.26	17.29	0.60	95	11100
18	4.457	1874.11	18.04	0.78	90	3700
19	3.892	1005.26	17.81	1.09	86	3700
20	3.282	1166.49	17.61	1.54	91	3700
21	3.843	1254.73	17.71	0.56	88	7400
22	3.432	1756.49	17.62	0.10	82	7400
23	4.304	1574.67	17.16	0.51	92	7400
24	4.388	1634.55	17.61	1.33	89	3700
25	3.919	1224.01	17.44	1.36	88	3700
26	3.577	1126.58	17.40	1.38	90	3700
27	4.492	1204.59	17.35	1.11	89	7400
28	4.054	1038.78	17.54	1.26	83	7400
29	3.350	1892.72	17.09	0.96	87	11100
30	3.430	1920.24	17.99	1.57	87	7400

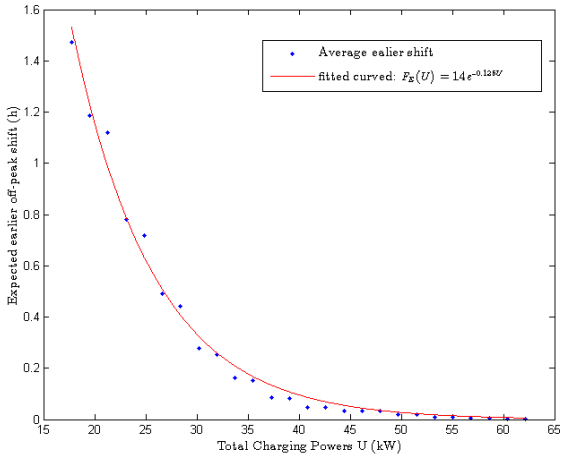


Figure 6-11 – Expected shift (h) to start before off-peak hours to assure feasibility according to power bandwidth used ($n_{EV} = 30$ and $n_{tests} = 10000$)

Finding the optimal total power bandwidth: Let the subscription fee per kW be equal to 19.5 euros, the difference price to pay for the standard and the off-peak tariff is 0.029 per kWh [22]. The optimized bandwidth to reserve according to (6.1) and (6.2) is 19.5 kW, with an expected daily earlier starting time of 1h. The power BW chosen is equivalent to 11% of the total 177.6kW charging power of all EV.

The required power bandwidth and the expected earlier shift behaviors according to random data: Given the specific configuration of the case study: gamma distribution random demands, normal distribution arrival time and fixed common departure, one can see specific behaviors of the power bandwidth and the expected earlier shift. The power bandwidth required for a feasible schedule within the off-peak hours follows the normal distribution with adequacy. The fitted function $F_E(U) = 14e^{-0.125U}$ of the expected earlier shift reflects confidently the data. According to those statistical results, the behaviors of the system can be predicted at any power bandwidth rate. For example, in this case, to cover 99% of the schedule feasibility over a large number of charging events, one has to choose a power bandwidth greater or equal to 41.6 kW. At that rate, the expected earlier shift to assure a 100% feasibility is 4.6 minutes. According to that analysis, the EV charging planner can make more sophisticated decision in the case of multi-criteria has to be taken into account.

6.5 Online scheduling

6.5.1 From deterministic scheduling to online scheduling

The previous section helps us to answer to the two decisions about the power subscription and the daily starting time of the charging procedure. However, the execution of the charging coordination is subject to more uncertainty: the random plugging times of the user and the daily load demand. In this section, we would present how one can deal with those unpredictable events while assuring the performance of the schedule (the total weighted completion time minimisation). First, we introduce briefly the online scheduling.

The online scheduling problem is a particular type of scheduling problem when the available information (problem input) becomes available continuously with time. The scheduling algorithm can only calculate a solution with this limited given amount of information, then updates the solution each time it gets new data. The way the scheduling algorithm update the existed solution denotes its ***decision making policy***. Since an online scheduling often required a rapid response for each new information, the scheduling algorithm implemented with the online scheduler should be a fast one. Any deterministic algorithm can serve as the solution procedure for the online schedule problem.

6.5.2 The decision making policies

There exist many policies to face the uncertainties. However, we try to group the decision-making policies into four categories based on the work of Pinedo [60]. The tasks are whether preemptive of non-preemptive. Also, there are two manners to update the baseline schedule: static list and dynamic.

In the static list decision policy, one calculates the schedule baseline only once, at time zero. Then, according to the schedule, one can create a priority list, called static list. Every time a task is completed, the next task will be executed next according to that static list. However, what happens if the task on top of priority queue has not arrived in the system yet? If tasks are preemptive, then an available task in the system which has the most priority can start without waiting. The scheduler will interrupt these tasks as soon as

a task in a higher position on the static list arrives. This policy is called *preemptive static list*. If the task is non-preemptive, then the scheduler will wait until the job having priority arrives at the system. This policy is called *non-preemptive static list*.

In the dynamic decision policy, the schedule baseline will be updated each time the scheduler get newly available information. For that reason, it requires a fast algorithm to keep up with the on-line event. If the tasks are preemptive, the whole schedule will be recalculated completely each time the system encounter a new event. On the contrary, is the tasks are non-preemptive, only the tasks which are available in the systems, yet, have not started are involved in the rescheduling. Hence we call this policy the *partial rescheduling*. We classify those policies in Table 6.2.

Table 6.2 – The decision making policies in resume

	Static list *The decision maker schedules all the jobs at time zero, then the priorities of execution of the jobs are fixed in a static list.*	**Dynamic** *Every time there exist a new information; decision maker may recalculate the schedule and reorder the task*
Non-preemtive tasks	In this policy, the scheduling baseline is unchanged during the execution. For example: if a job i has the priority to be executed, but i has not arrived on the system yet, the system will put the machine in idle to wait for the job.	This policy is called partial rescheduling. Only the tasks which have not been scheduled yet are rescheduled. Once a job has been started, it would have the priority to end before the decision maker schedules others.
Preemptive tasks	This is a no-wait policy when a job has a higher priority in the list has not arrived in the system, the next job can be executed instead. However, as soon as the privileged job arrives, this executing job must be interrupted to make a place for the privileged one and then resumes to process later.	This policy is called complete rescheduling. All the tasks are rescheduled every time the decision maker has new information; he can interrupt processing tasks if necessary.

6.5.3 Reactive rescheduling

Since the tasks are non-preemptive, then the schedule reparation should be partial. Only jobs in the waiting queue are about to be rescheduled, the tasks which have already

started keep on charging, and being a constraint for the schedule update. The detailed procedure is displayed on figure 6-12. The online scheduler starts at the "time-zero" decided by the simulation based forecast. Then, it schedules the jobs already available in the system (plugged EV). The charge point executes the resource allocation according to the schedule, and wait for new events. For each new event, the scheduler modifies the set J' of available jobs in the system and the set Ω of executing jobs. According to the partial rescheduling policy, the deterministic algorithm then schedules the new input $J' \setminus \Omega$ with subjects to the set of Ω executing jobs. A new schedule for jobs in the queue is established, and the online scheduler executes it.

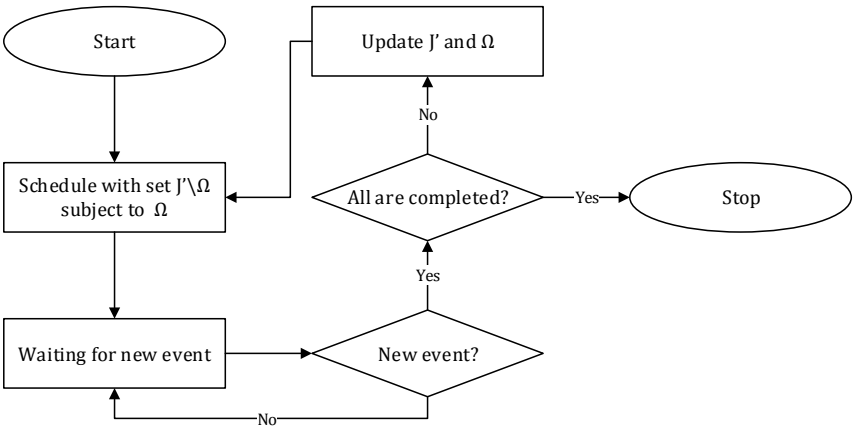


Figure 6-12 – The online schedule policy

Figure 6-13 illustrates a schedule example. The decision maker is launched daily to decide the starting time of the charging process which is, in this example, earlier than the starting time of off-peak duration. The online scheduler starts scheduling with available tasks at time zero. Then it encounters a new arrival of EV i. This arrival modifies the set J' of available jobs, hence: $J' := J' \cup i$. The schedule is then modified by the algorithm with the new input J'. The scheduler waits for the new event which is, in this example, a job being completed earlier than planned. In this case, only the set Ω is modified, and the algorithm partially updates the schedule.

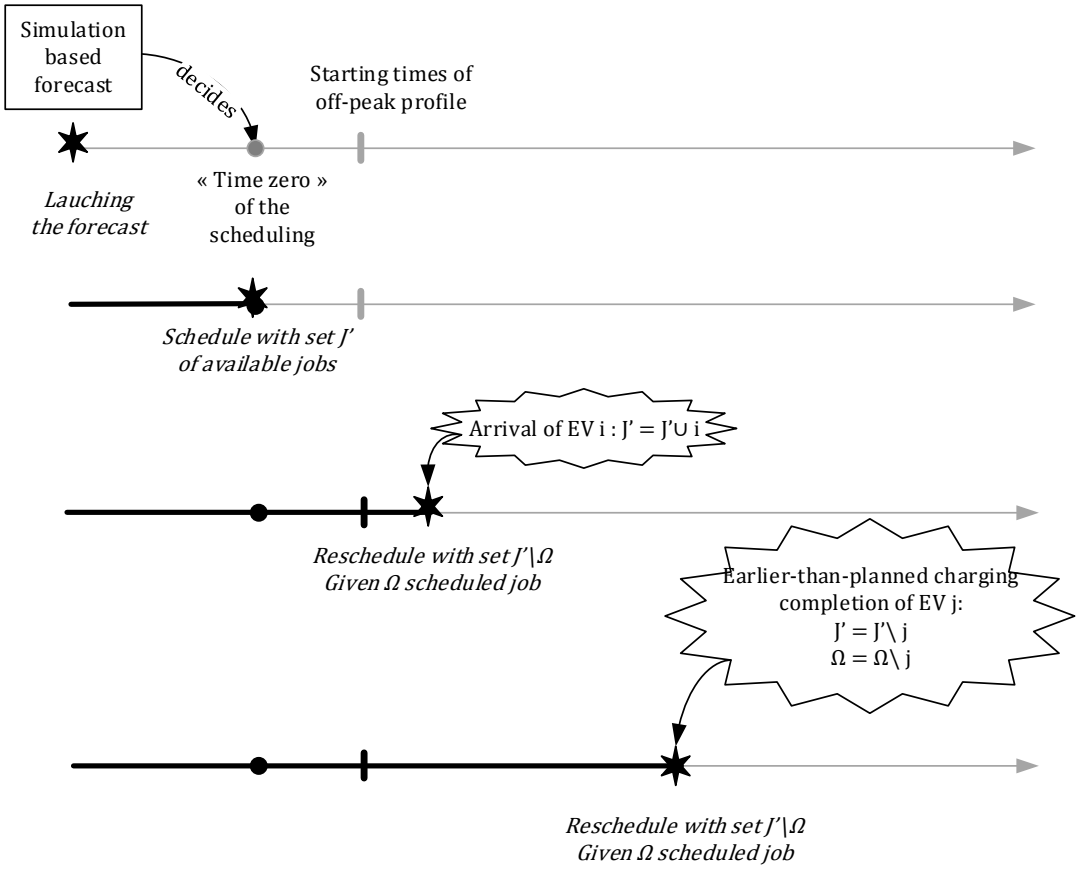


Figure 6-13 – The online schedule policy in example

6.6 Conclusion

In this chapter, we provide the method to implement the efficient deterministic algorithms developed in the previous chapters to the real-life problem. The scheduler has two main components: the forecast (proactive) schedule and the reactive rescheduling. In the first part of the chapter, we presented the theoretical background of the method. Then, we introduced the conception of the simulation based forecast model. This forecast model can take advantage of the given degree of knowledge we get from the database: the past behaviours and demand of the EV customers on the parking. Hence, with the random distribution of the two possible disturbances on the arrival times and the daily electrical demand of the battery, the simulation based helps us to observe the randomness behaviours of the total power demand for the charging process with a given deterministic scheduling

algorithm. Thence, with that stochastic property, one can estimate the total charging power to subscribe for the parking and the daily starting time of the charging procedure.

The rest of this chapter was dedicated to the presentation of the online scheduling policy. Taking into account the characteristic of the charging task and the performance of our scheduling algorithm, we developed a partial rescheduling policy for the online scheduler. While the new information reveals in time, the scheduling algorithm must incorporate in permanent with new changes and update the rescheduling partially. It, at the turn, avoids changing the task currently in charge (non-preemption) and takes those scheduled tasks as the constraints of the new EVCC scheduling.

In the next chapter, we will introduce the implemented software which concretises the conception developed in this chapter.

The work done in this chapter is subject to two publications [53, 54].

Chapter 7

Software implementation

7.1 Introduction

This chapter is dedicated to introduce overall the stand-alone scheduler software implemented on the parking charging back office. The thesis has started with an industrial problem; hence it should be finalised with an industrial implementation. Concretely, a scheduler software has been developed. In this chapter, we present first the overall condition of the electric charging service solution at **Park'nPlug**. Then, we introduce the definition of the necessary components of the solution. After that, we present the software design. At the end of the chapter, we conclude on the implementation and discuss our future developments.

7.2 The EV charging management description

The charging management has two parts: the parking (local management) and the information system (global management). We present first the principal components of the local management:

The NEMO box is the Back Office of the parking. It collects information from watt meter and EVSE, giving the command to each charge point about the authorisation/interruption of charging and changing the allocation of electrical power...

The Keyboard and Badge reader identifies the client of the parking. This authentication gives the NEMO information about the customer hence can bill the charging cost properly.

The EVSE supplies the power to the EV charger.

The watt meter which measures the power of each EVSE. It connects directly to the NEMO Box to provide real-time information about the charging power.

The electric supplier watt meter measures in real-time the power bandwidth of the total available power dedicated to the charging procedure. It also communicates in permanent with the NEMO.

The relay is used to control the EVSE: turn on or turn off an EVSE. Furthermore, it can control the power flow to each EVSE.

The modem permits the NEMO to communicate with the server, hence bridging the local to the global management.

In the global management (the information system), we have a management server which communicates with the NEMO box. This server collects information and data on the local parking to let the administrator keep track of any remote events. It also visualises the data for the customers and let the clients, in their turn, sending their request, for example, an immature interruption or fixing another departure time. Figure 7-1 provide more details on the local and the global management of the parking.

7.3 Software design

Given a limited bandwidth of communication between the NEMO box and the server, the algorithm has to be implemented locally, on the microprocessor of the NEMO Box. This microprocessor is a *Raspberry Pi 2 Model B* having 900MHz quad-core ARM Cortex-A7 CPU and 1GB RAM. First, we define the components used in the software implementation:

Operational Management (GO) The management software of the NEMO Box

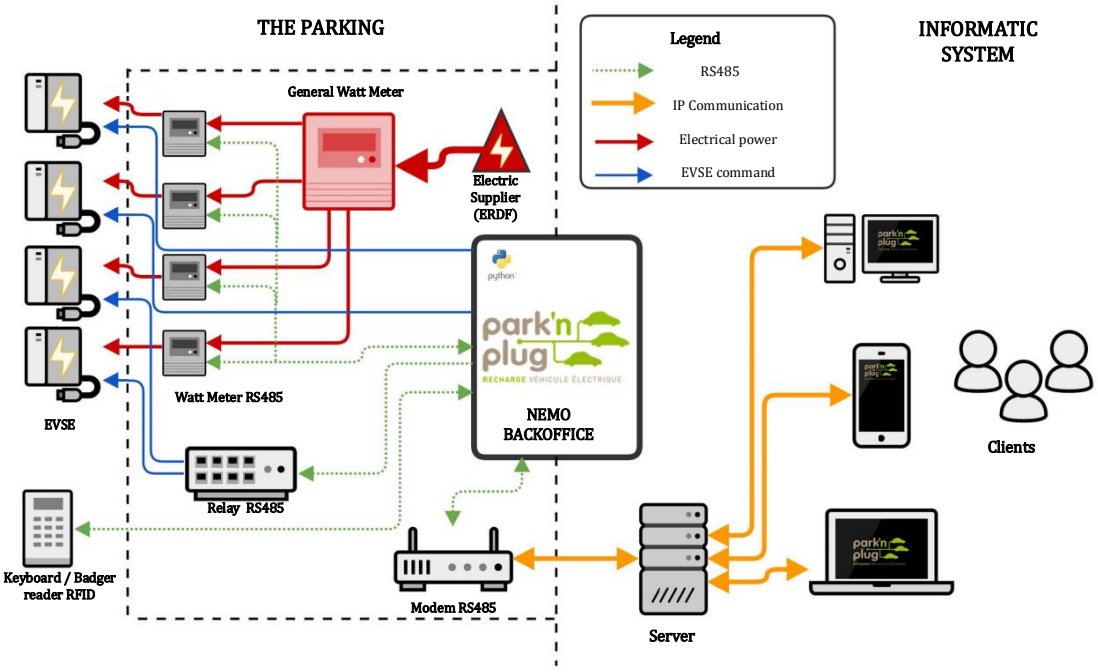


Figure 7-1 – The schematic of the charging solution. *Source: ParknPlug*

Algorithmic Platform (PA) Platform on which the Algorithm Box and Operational Management are implemented. It also ensures good inter-procedures communication and manages fail-safe.

Algorithm Box (BA) Box contains solution procedure algorithms. It makes the choice of scheduling method automatically after each request of the Operational Management. There is two components:

- Library of schedulings algorithms (BIB-ALG). The set of methods implemented to respond to the various current situations. Current version: BIB-ALG 1.0 with the Heuristic of Layering and Adapting (HLA), the Greedy Best Fit algorithm (GBF) and the Reserved GBF
- The Simulation based forecast module (MOD-PREV). The module processes the NEMO database (NemoDB) to the historical database (HistDB).

Communication Management Unit (CMU) The module by which the NEMO box can communicate with the BA by a predefined event set and translates the scenario received by BA for the control of the NEMO terminals.

Figure 7-2 details the information flows between the components.

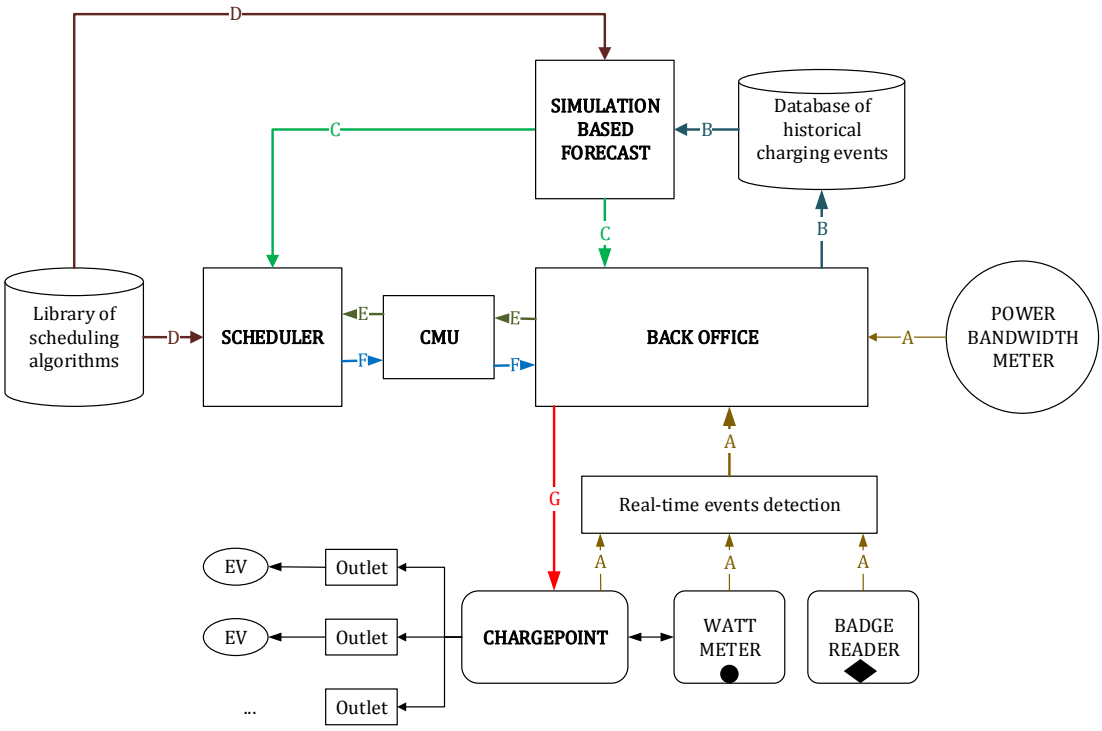


Figure 7-2 – The information flow

We explain each flow of information in the following list:

Flows A The real-time events flows which gives the NEMO information in permanent about any new event available. The Badge reader tells the NEMO about the new arrival of charging task. The wattmeter tells the NEMO about the current charging power of each job. The charge point tells the NEMO about its state and availability. The power bandwidth meter keeps the NEMO be informed about the power bandwidth. All the real-time events are treated by a specific module before the information goes to the NEMO to avoid data overloaded.

Flows B The flow of stocking events. For each charging transaction, the NEMO stocks all the logs (arrival times, charging duration, charging demand...) on its database. The Simulation-based forecast (MOD-PREV) can request information from this database to make decisions and predictions.

Flows C Forecast flow tells the back office when to start the charging process, and estimate

values for the schedulers.

Flows D Algorithm flow gives the scheduler and the forecaster the suitable optimisation from its database. For example, the forecaster may request the reversed GBF while the scheduler requires the HLA.

Flows E Solution request flow: the NEMO request new algorithm from the Algorithm Box to respond to the new situation. This request is stocked first in the MCU to avoid overloading the calculation of the BA.

Flows F Solution respond flow: the BA responds the new solution for the NEMO. The CMU also manages this flow.

Flows G EVSE control flows: the NEMO controls the EVSE using this communication channel.

The PA is created to assure at the same time a good and flexible rescheduling system with a tight synchronisation to the electronic components managed by the GO. The synchronisation between the GO and the BA is driven by the CMU, detailed in figure 7-3. For every unexpected event captured, if the BA is not actually busy, it requests the new updating on the schedule. Otherwise, it stacks all the changes and then send new information as soon as the BA becomes available.

One can see that Flow A triggers the scheduling. Hence, by analysing the information of the sensors (watt meters, charge points, badge reader...) the real-time event management finds a suitable event defined in the **EventList** (table 7.1 and table 7.2) to send to the GO. The GO stocks new developments in the database throughout the Flow B, then in the same time, the MOD-PREV sends new estimation for the scheduler by using flow C. In the same time, if the NEMO does not have an available scenario to respond to this new event, it has to use the Flow E to demand a new schedule from the BA. The BA then responds with anew solution through the Flow F. After having the new scenario; the NEMO would control the EVSE according to this new schedule.

Table 7.1 presents a partial **EventsList** of the NEMO.

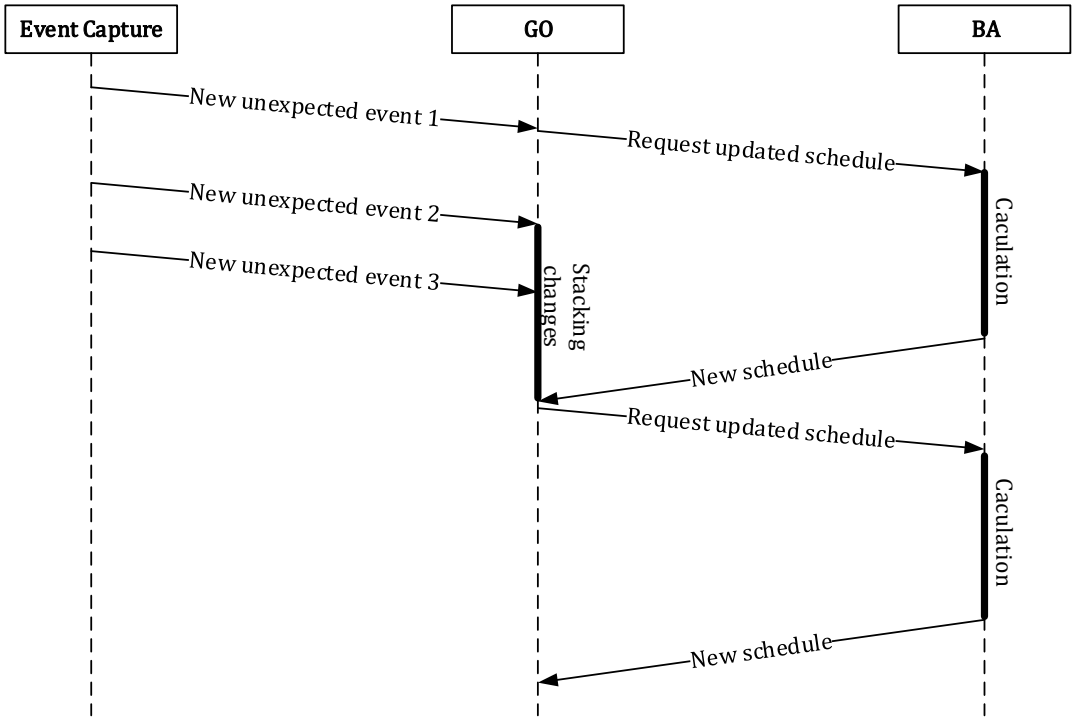


Figure 7-3 – The CMU description

7.4 Conclusion

The thesis starts with an industrial problem and ends with an industrial implementation. The software design transfers the algorithms designs and all the scientific development into a deployable product. We have deployed the first version of the software PA 1.0, BA 1.0 and MOD-PREV 1.0 which can cope with the ACPF 1/2 problem. The deployment tests are very positive: the performance of the BA is assured. It guarantees a response in less than 20 seconds when the size of tasks goes up to 50 EV. The future works on the software are promising: a more sophisticated EventList for the next version of the software could cope with the ACPV 2.

Table 7.1 – Event definitions

No.	ID	EVENT	Sensors	Technical definition
1	AUT	Authentication	Badge reader	Plugging and authentification of the EV users
2	BOC	Starting of a cooling period	Watt meter	A cooling period which consumes constantly a little amount of electrical power is considered to be the end of a charge.
3	FDC	End load	Watt meter	If present consumption of less than DM threshold for a duration ITC and there is a higher consumption prior to SM
4	ADC	Charging cancellation	Badge reader / GO	End-of-charge not caused by CNU or FDC: Cancellation by user or management Operationelle
5	DCF	Starting of forced charging	Watt meter	A planner starting a forced charging (a charge that takes effect immediately without being scheduled)
6	FCF	Ending of forced charging	Watt meter	A planner ending a forced charging
7	PEA	Consumption less-than-set	Watt meter	Actual power received by EV lower than that configured for a duration ITC
8	CNU	Null charging	Watt meter	If present consumption below SM ITF threshold for a time without previous consumption
9	SHU	Shutdown the BA, FAILSAFE mode	None	Shutdown the server BA, then the use the FIFO failsafe planning

Table 7.2 – Events notations

Notation	Definition	Unit
SR	Minimum consumption threshold above which, and after an ITR time, consider a cooling period	Wh
SM	Lower consumption	Wh
ITR	Cooling period test interval	s
ITC	Consumer test interval	s
WDI	Watchdog interval defined as no response from BA	s

Chapter 8

Conclusion et Perspectives

In the context of the sustainable development and the utilisation of clean energy, the research on the electric vehicle technologies has gained much attention. More than ever, we face a rapid growth in the sale of electric vehicles. Hence, the research on the management of the charging of the EV carries a great importance. By the same reason, the optimisation of the EV coordination is in constant development and concurrence. In the middle of the current global research wave, this thesis tends to propose a measured solution for the EV charging coordination problem within the specific context of France. The work done in this thesis is the fruit of an Industrial Convention with the partnership of the company **Park'nPlug** based on Rosières-près-Troyes, France and the **Laboratory of Industrial System Optimisation** of **UTT**, France. This company provides the solution for the EV charging management of the private parking. To take the lead in the novel technology, a research of a "Smart Charging" can assure an optimal schedule and an autonomous functionality which also are the objectives of the research of this thesis.

We resume the contributions made throughout this thesis as follows:

- **First, the state-of-the-art pointed out the research gap on a scheduling problem, where the tasks are malleable (or semi-malleable) with the consideration of a time-varying resource profile. By that analysis, we can po-**

sition our research in this thesis to fill the specific gap. The methodology was appropriate which can provide us with a solid base for all the developments. The definition of the new problem class, with the identification of the EVCC constraints, tackle the real-life problem directly. Also, five configurations: ACPF 1/2 and ACPV 1a/1b/2 helps us to classify the existed industrial problem. The complexity analysis of each problem according to the standard optimisation objective suggests the suitable solving method corresponding to each configuration.

- With a scientific approach, we analysed different techniques to formulate the ACPV 2, the most generic and challenging problem to solve among the five configurations. The two chosen techniques were to formulate the problem in a conjunctive ($P-Cml$) and a disjunctive ($P-Dsj$) ways. A full analysis on the convex hull and the LP-relaxation leads to the introduction of the two cuts for the $P-Dsj$. The numerical tests verified our theoretical analysis on the two formulations and stated the strongest formulation is the $P-Dsj$ with a single cut 1.

- Since the $P-Dsj$ formulation is in a block matrix structure, it helps us to decompose the problem by the Danzig-Wolfe principles. This decomposition resulted in the construction of the Branch-and-Price Algorithm to solve the problem. The numerical tests proved that the strength of the algorithm relies very much on good branching strategies. Also, the Branch-and-Price based on the column generation algorithm, so, it requires proper stabilisation techniques. We provided an efficient branching policy for SOS1 variables and apply the novel dual bound to stabilise the column generation procedure. In search of a faster resolution method, a family of constructive heuristics is presented. The heuristic is proven to be dedicated to the ACPV 2 problem. It proves an excellent performance and rapid response to the real-life size problem.

- By the end of this thesis, we design a predictive-reactive rescheduling method that can cope with uncertainty and real-time event. By that mean, one can bridge all the theoretical work into an implementable application.

The simulation based forecast (predictive scheduling) helps to stabilise the electrical grid by reducing the diversity factor of the charging parking to less than 20% according to our case study. It also optimises the power usage cost by deciding a good power subscription for the parking and daily starting time for the charging procedure. The predictive-reactive scheduling is concretised by a deployment of a stand-alone scheduler software. The structure of the software is designed to be compatible with the existed hardware and software condition of the company and prove a smart-charging function to the charge point. This software is a competitive advantage for the company, where we were one of the first in France to provide an EV smart charging service.

Before ending this dissertation, we would state our perspectives, point-by-point according to each contribution.

First, the study of the formulation should include a more general case, where the tasks are permitted to have a fixed k interruptions. This formulation would improve the objective value while maintaining an acceptable margin of battery lifecycle security.

Second, a diving heuristic should be designed to make the branch-and-price algorithm being more dedicated to the scheduling problem. The branch-and-price are generic; we could modify the local constraints without re-designing a new algorithm. So, a faster Branch-and-Price should open more possibility to solve many different derivative problems from the ACPV 2.

Third, the actual HLA is till based on elements consisting of randomness. We are currently researching for an adaptive search method, which can improve the efficiency of each shaking (destroy and construction) of the solution. Also, the complexity of the HLA depends greatly on the complexity of the GBF (the initial solution construction phase). Hence, a more efficient restricted search would reduce the searching space of the GBF.

Fourth, we consider the construction of a robust scheduling problem, since the random distributions of the inputs are known.

Finally, an improved version of the current software which can consider more sophisticated events is also a future work.

Appendix A

Résumé étendu en français

Optimisation de l'ordonnancement de recharge des véhicules électriques

A.1 Introduction generales

A.1.1 Motivation et contexte industriel

Ce travail de thèse a été réalisé dans le cadre des Conventions industrielles de formation par recherche (CIFRE) avec la coopération entre le **Laboratoire d'optimisation du système industriel** (LOSI) et l'entreprise **Park'nPlug**. Il est donc important de souligner le contexte industriel qui motive et oriente dans le même temps le centre d'études au cours de cette thèse. Le problème posé dans cette thèse est l'optimisation de la planification de recharges des VE, qui est également connu sous le nom de problème CRVE. Au cours des dix dernières années, nous avons observé un grand bon technologique des véhicules électriques. La densité de la batterie augmente de 400 % en 7 ans (2008-2015) alors que le coût par kWh est réduit de 73 % [17]. En conséquence, le nombre de voitures électriques, inférieures à mille en 2005, s'élève à 1,26 million en 2015 [17]. L'initiative Véhicules électriques (EVI) définit un objectif de 20 millions de VE déployés d'ici à 2020. Au milieu de la crise écologique et énergétique, un changement des véhicules à énergie fossile vers des véhicules électriques devient une solution durable permettant de réduire les émissions de carbone et d'assurer la sécurité énergétique. Ainsi, supporter le succès du déploiement VE c'est la politique avantageuse des gouvernements et la prise de conscience de l'utilisateur de VE sur le développement durable. Toutefois, la charge de VE est une énorme tâche de consommation d'énergie sur une période prolongée. Pour le PDC monophasé, la puissance nécessaire peut varier de 3,3 à 7,4 kW pour une durée de charge totale de 3 à 6 heures. Un chargeur triphasé peut quant à lui fournir de 10 à 22kW pour 1 à 3 heures de charge [19]. Surtout, il existe également un mode de charge ultra-rapide avec une puissance de 43kW. Par conséquent, la forte pénétration des véhicules électriques peut causer de nombreuses préoccupations au réseau électrique. En plus, l'abonnement de puissance pour le PDC individuel serait un gaspillage, car le temps de chargement prend en moyenne 17 % du temps de disponibilité. Un PDC personnel et son coût électrique pourraient être coûteux pour l'utilisateur du VE. En tant que bonne solution à ces problèmes, un processus de recharge collectif et coordonné peut stabiliser la grille et optimiser la productivité des points de charge, en minimisant également le coût électrique.

La société **Park'nPlug**, située en France, fournit les services et la gestion de recharge

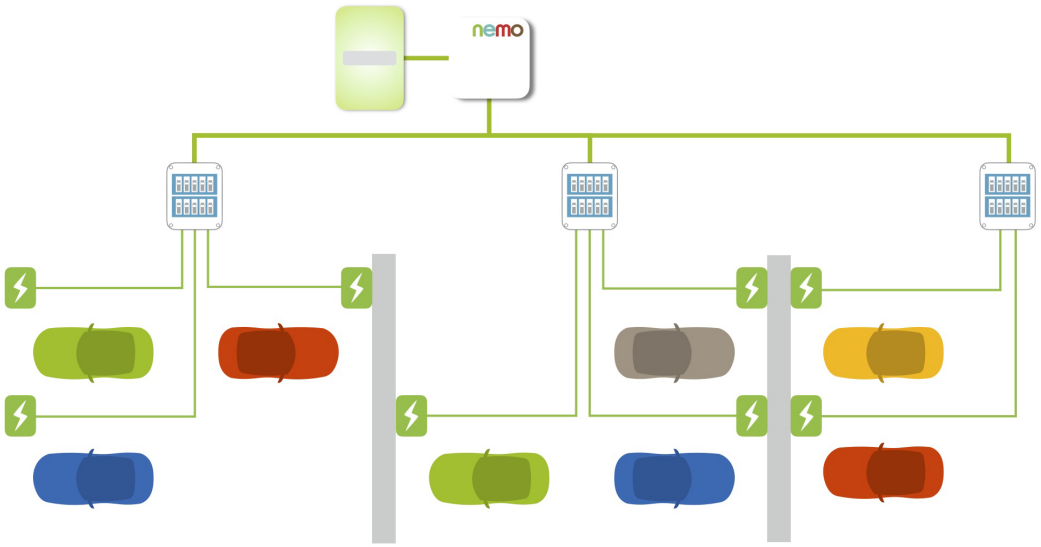


FIGURE A-1 – Une illustration de stationnement des VE en charge. *Source : ParknPlug*

des VE en milieu collectif. En France, le secteur privé domine avec une part de 85% des points de charge par rapport à 15% du secteur public (supermarché, public, stationnement routier) [17]. La principale différence entre ces deux domaines est le niveau de connaissance des données d'entrée. Dans le parking privé, nous connaissons tous les clients et leurs comportements passés alors que le client d'information du secteur public est totalement inconnu. Par conséquent, la méthode d'optimisation de la planification de charge serait complètement différente face à chacun des deux problèmes. Dans la limite de cette thèse, nous abordons exclusivement la coordination de charge des VE du secteur privé. Le coût électrique est comparable à tout autre coût dans le problème de planification : il consiste en un coût fixe et un coût variable. Le coût fixe est un abonnement mensuel et concerne la bande passante totale. De nombreux fournisseurs d'électricité français proposent deux plans de tarification : si le coût de l'énergie est constant ou s'il comporte deux plages, le coût d'heures creuses (moins cher) et le coût d'heures pleines (plus cher). Avec la première plage, le prix constant, nous avons peu de possibilité d'optimiser le coût variable. Néanmoins, la deuxième plage de tarification nécessite une planification de recharge optimale. Il faut faire face au compromis entre le coût fixe et le coût variable. Un abonnement de puissance étroit est bon marché, mais, prend plus de temps pour charger tous les VE. D'autant plus que ces derniers peuvent prolonger le temps de chargement sur les heures de pointe, augmentant ainsi le coût variable. Ainsi, la minimisation du coût électrique devrait répondre à deux questions :

FIGURE A-2 – Un VE en charge. *Source : ParknPlug*

1. A quel abonnement électrique devrait-on souscrire en milieu collectif pour minimiser le coût fixe ?

2. Quand devrions-nous commencer la charge pour minimiser le coût variable.

Après avoir répondu à ces questions, nous passons à la phase suivante : l'optimisation de la qualité du service. On devrait libérer dès que possible toutes les tâches de recharge dans la file d'attente, en ce qui concerne toutes les contraintes concernant la disponibilité de VE sur le stationnement, la demande de charge quotidienne et les références de départ des clients.

A.1.2 Contributions

La première contribution de cette thèse repose sur la création d'une méthodologie pour résoudre le problème industriel : une classification sur les contraintes CRVE et la définition d'une nouvelle classe de problème d'ordonnancement sous une seule ressource supplémentaire. Cette classe de problème d'ordonnancement est exprimée en 5 configurations à savoir ACPF 1/2 et ACPV 1a / 1b / 2.

La deuxième contribution est la formulation du problème CRVE. Nous proposons deux techniques de formulation, suivies d'une analyse détaillée de leur LP-Relaxation.

La troisième contribution principale consiste en deux méthodes de résolution du problème CRVE. Nous proposons la procédure de solution exacte par l'algorithme de branchement et de prix. Nous présentons avec cette approche l'outil de stabilisation numérique et les stratégies de branchement. En outre, nous concevons une famille d'heuristiques constructives HLA, basée sur les principes de l'"*Emballage de la Bande*" et la "*Recherche de Grande Voisinage*", ce qui permet de résoudre efficacement les grandes instances du problème dans un très court délai d'exécution.

La quatrième contribution est la conception d'une prévision basée sur la simulation et d'un planificateur en ligne pour faire face aux incertitudes du calendrier et aux événements en temps réel. Il est adapté à la mise en œuvre technique grâce à sa reconfigurabilité et facilité d'utilisation.

Enfin, nous déployons toute la recherche algorithmique effectuée en créant un logiciel de planification. Ce logiciel est autonome et implémenté dans un microprocesseur fournissant l'intelligence de charge au PDC

A.1.3 Contenu de la thèse

Dans ce premier chapitre, nous soulignons la motivation et le contexte de la thèse. Pour fournir une base théorique pour notre développement scientifique, nous décrivons dans le chapitre 2 les terminologies et les définitions des problèmes d'ordonnancement avec une ressource supplémentaire, le problème de coordination de la charge du véhicule électrique et les techniques de formulation de PLMNE. Nous examinons les travaux existant et identifions la manque de recherches clé afin que nous puissions positionner notre recherche.

Dans le chapitre 3, pour fournir la méthodologie adéquat, nous identifions les contraintes de coordination des véhicules électriques, selon cinq ensembles de contraintes introduisant aussi cinq configurations. Dans le reste du chapitre, nous formulons la configuration la plus générique de la classe de planification, la formulation ACPV 2 en deux PLMNE : une formulation conjonctive et une formulation disjonctive. Pour avoir une connaissance plus profonde du problème, nous faisons une analyse convexe des deux formulations et introduisons deux coupes valides. Nous testons à la fin du chapitre la résolution exacte de ces deux modèles

développés.

Le chapitre 4 et le chapitre 5 sont dédiés à la procédure de recherche de solution au problème. Dans le chapitre 4, nous déclarons que la formulation disjonctive du problème consiste en une matrice partitionnée. Pour cette raison, nous pouvons appliquer la décomposition de Danzig-Wolfe pour partitionner la formulation. Pour cette raison, nous présentons l'Algorithme Branch-and-Price comme une approche exacte pour résoudre le problème CRVE. À la recherche d'un algorithme plus rapide pour faire face aux instances de plus grande taille, le chapitre 5 abordera la conception des heuristiques de stratification et d'adaptation dédiées pour résoudre le problème. Toutes les méthodes de résolution sont testées numériquement afin que nous puissions étudier leur force de calcul.

Pour faire face aux incertitudes temporelles, nous présentons tout d'abord dans le chapitre 6 une prévision basée sur la simulation. Le prévisionniste décide de la quantité d'énergie électrique que l'on doit souscrire au niveau stratégique. De plus, il fait preuve de précision quand faut-il commencer la procédure de facturation quotidienne. En outre, nous présentons la conception d'un planificateur en ligne. Cette conception apporte un algorithme déterministe pour faire face à l'événement en temps réel par la politique de ré-ordonnancement partiel.

Dans le chapitre 7, nous mettons en avant la recherche scientifique sur la conception d'un logiciel de planification autonome déployable. Ce logiciel est implémenté sur le microprocesseur à lq centrale de gestion de charge des VE.

À la fin de la thèse, nous reprenons les résultats de nos travaux, tirant ainsi des conclusions. Enfin, nous pouvons exposer la perspective et les travaux futurs comme point de vue de cette thèse.

A.2 L'état de l'art

A.2.1 Introduction

A.2.2 CRVE

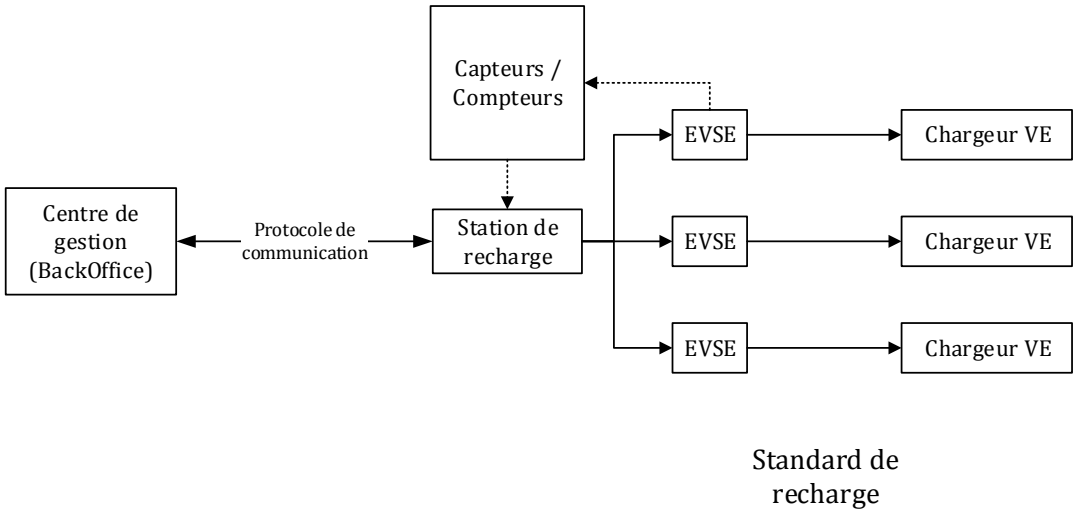


FIGURE A-3 – L'infrastructure de chargement simplifiée

Revue et classement de la littérature Il existe deux approches majeures pour résoudre le problème de coordination VE existant dans la littérature que nous pouvons définir l'approche locale et l'approche globale. Pour l'approche locale, la procédure de solution est implémentée localement, habituellement au point de charge. La décision est décontractée par une règle de base (décision de type "quoi faire"). La méthode peut réagir rapidement et en temps réel grâce à sa simplicité. Cependant, comme le problème d'entrée n'est pas entièrement traité, la performance globale de l'ordonnancement n'est pas suffisante. La deuxième procédure de résolution du problème CRVE est la méthode de planification globale. Cette approche est appelée globale car elle prend l'entrée dans son ensemble. En utilisant cette approche, les solutions trouvées peuvent avoir une meilleure performance par rapport à la solution obtenue avec la méthode locale. Toutefois cette méthode trouve des difficultés à faire face aux incertitudes.

Méthode de planification	Approche de résolution	Travaux	Objectifs		
			Minimisation de coût	Nivellement des ressources	Minimisation des critères de temps
Ordonnancement local	Algorithme flou contrôlé par rétroaction	[48]		x	
		[23]	x		
	Contrôleur basé sur la tension	[4]		x	
	Contrôleur décentralisé	[6]		x	
Ordonnancement global	Algorithme génétique	[5, 46]		x	
	Optimisation des essaims de particules	[59]	x		
	Gestion de la demande	[2]		x	
	Ordonnancement stochastiqueg	[35]	x		
	Emballage de la bande d'alimentation	[44]		x	

TABLE A.1 – Classification de certaines approches CRVE dans la littérature

Nous reprenons tous les travaux concernés dans le tableau A.1.

A.2.3 Problème de planification avec une ressource supplémentaire unique

A.2.4 La revue de littérature

Nous reprenons dans table A.2 la classification des travaux concernés dans cette section avec leurs contraintes et objectifs correspondants, ainsi que la procédure de solution.

Approche	Méthode	Travaux	Ressource constante	Ressource variable	Temps d'arrivée	Date limite	Objective
Problème d'ordo. discret-continu	Recherche tabou	[78]	x				Minimisation de makespan
	Programmation dynamique et heuristique	[65]	x				Somme des temps de présences pondérée
	Algorithme du temps polynomial	[67]	x			x	Le nombre pondéré d'emplois tardifs / makespan / coûts totaux de consommation de ressources.
Problème de planification cumulative	Algorithmes d'ajustement temporel	[7]	x		x	x	Tests de satisfaction
	Réglages des têtes (planification pseudo-préemptive de Jackson)	[16]	x		x		Minimisation de makespan
Problème d'ordo. des tâches rigide	L'algorithme Greedy Best Fit	[18]	x				Minimiser le nombre de machines
Problème d'ordo. de tâche moulable	Algorithme sous-optimal	[14]	x				Minimisation de makespan
Problème d'ordo. de tâche malléable	Classe dominante de planification	[64]	x				Temps total d'achalandage pondéré
	Algorithme d'emballage des rectangles	[13]	x				Minimisation de makespan

TABLE A.2 – Classification de certaines approches en vedette sur le problème de planification théorique sous contrainte de ressource

Les travaux existants sur CRVE ont repris dans le tableau A.2 soulignent l'absence de méthodes de résolution avec l'optimisation des critères de temps. En outre, une planification complète et globale qui considère toutes les contraintes possibles n'a pas encore été développée. Il existe encore un vide concernant la contrainte de ressources variant dans le temps. De plus, selon notre connaissance, il n'y a pas de problème générique et complet, qui tienne compte de toutes les contraintes suivantes : la restriction temporelle de la tâche, le profil de ressources dépendant du temps / constant, la consommation de ressources de la tâche semi-continue. Par conséquent, nous positionnons nos recherches pour combler cette lacune essentielle dans la littérature.

A.3 Classification des problèmes CRVE et Formulation mathématique

A.3.1 Classification des problèmes CRVE

En passant en revue la littérature, on peut trouver les deux principaux inconvénients des approches globale pour résoudre le problème de CRVE. Le premier est le manque de contraintes temporelles et l'optimisation des critères de temps. Le deuxième écart d'étude des approches mondiales est l'absence de méthode de résolution rapide. En outre, il manque un modèle générique qui peut faire face à différents problèmes avec des contraintes similaires. Étant donné que notre objectif industriel est de développer une solution de recharge pour le domaine résidentiel, nous avons deux priorités. La première est la faisabilité de la planification du comportement des utilisateurs, y compris les contraintes temporelles de stationnement et de temps de départ, ainsi que la demande quotidienne d'énergie. La deuxième priorité est d'assurer la qualité de service ; alors tous les VE doivent être complètement chargés dès que possible. Ensuite, le choix d'utilisation de l'approche globale est la meilleure axe de recherche. Pour compléter le résultat de recherche de cette méthode, nous souhaitons formuler une famille de configurations CRVE avec des contraintes et des propriétés particulières. Avec ce type de classification, on peut décider quels travaux de recherche peuvent être utilisés pour résoudre spécifiquement chaque problème.

Dans le tableau A.3, nous représentons différentes configurations avec leurs contraintes respectives. Au préalable, la complexité de chaque optimisation est citée. Nous présentons dans la dernière colonne les travaux qui peuvent être utilisés pour résoudre chaque configuration. Plus de détails sur la preuve de la complexité de chaque configuration seront précisés dans les chapitres suivants.

A.3.2 Formulation de PLNEM

Le problème ACPV 2 consiste en un profil de bande passante de puissance dépendant du temps. Le profil de puissance variable dans le temps à chaque intervalle de temps une valeur continue différente de la puissance totale disponible. En raison de ce profil, la formu-

Configurations	Sous-Configurations	Bande passante électrique		Puissance de recharge			Complexité			Algorithme d'ordonnancement
		Constante	Variable	Fixe	Continue	Semi-continue	Faisabilité	min C_{max}	min $\sum w_i C_i$	
ACPF	1	X		X			NP-Hard [18]	NP-Hard [14]	/	[14, 21, 33, 54]
	2		X	X			/	/	/	/
ACPV	1a	X			X		Polynomial [10]	Polynomial [13]	Polynomial [10]	[10, 13, 39]
	1b	X				X	NP-Complete (Fort) [7]	NP-Hard (Fort) [16]		[7, 10, 16, 64]
	2		X			X	NP-Hard [56]	NP-Hard (Fort) [56, 57]		[52, 56]

TABLE A.3 – The ACPF - ACPV configurations

lation du problème doit être discrète. L'horizon de planification contient des intervalles de temps de H (ou des intervalles de décision) indexés par k. La décision de ACPV 2 est exprimée sous deux aspects : la décision de temps et l'allocation de ressources. Concrètement, pour chaque travail, il faut décider quand commencer et finir, à chaque intervalle, combien de ressources peut consommer. En raison de cette raison, si l'on souhaite formuler le problème par la formulation de programmation linéaire, ce dernier devrait prendre la forme d'une programmation linéaire mixte avec la formulation indexée dans le temps.

Nous utilisons une approche standard pour formuler le problème ACPV 2. Compte tenu d'un ensemble de n tâches, $J = \{J_1, J_2, \ldots, J_n\}$ à programmer sur un horizon de planification des intervalles de décision de H, chacun d'eux dure Δt unités de temps. La notation $k \in \{0, ..., H\}$ indexe chaque intervalle de décision. Chaque emploi a une date de sortie r_i quand il arrive dans le système, et un délai d_i, quand il doit partir : $r_i \in \{0, ..., H\}$ et $d_i \in \{0, ..., H\}$. Entre deux temps d'intervalle $[k, k+1]$ le traitement du travail i prend en continu une quantité de ressources décidée par $u_{i,k}$. La ressource totale disponible à l'intervalle de temps k est U_k.

Fonction objective ACPV 2 vise à maximiser la qualité du service de chargement, puis nous introduisons la minimisation du temps d'achèvement pondéré total de l'objectif général $\sum_{i \in \mathcal{J}} w_i C_i$.

En résumé, nous présentons les variables et les paramètres suivants

Variables

- $x_{i,k}$: État du travail i au moment de la décision k
- $u_{i,k}$: Montant de la ressource allouée en permanence au travail i pendant l'intervalle k

Paramètres et données

- $\mathcal{J}$ ensemble des emplois

- $\mathcal{T}$ intervalle de temps
- H : horizon de planification.
- $\bar{u}_i$ et $\underline{u}_i$: capacité maximale et minimale du montant des ressources consommable du travail i
- r_i : temps de libération du travail i.
- d_i : date limite du travail i.
- e_i : début du temps d'achèvement du travail i.
- l_i : dernier début du travail i.
- h_i : l'efficacité de chargement des ressources du travail i.
- $\tilde{x}_i$: demande de ressources du travail i.
- x_i^* : demande de ressources normalisée du travail i.

A.3.3 Formulation cumulative (conjonctive)

L'idée qui diffère des deux formulations est la façon dont nous formulons le traitement et l'achèvement des travaux. Dans la formulation cumulative, le traitement du travail i est noté par $y_{i,k}$. $y_{i,k} = 1$ si le travail i est en cours de traitement au temps k, sinon, $y_{i,k} = 0$. La condition d'achèvement du travail est contrôlée par la variable binaire $c_{i,k}$ qui tourne à 1 seulement si le travail i a déjà atteint son état final au début de l'intervalle k $x_{i,k} \geq \tilde{x}_i$; Sinon $c_{i,k} = 0$. Avec cette variable donnée, le temps d'achèvement de la tâche i est :

$$C_i = H - \sum_{k=0}^{H} c_{i,k} + 1 \tag{A.1}$$

Le temps total d'achèvement pondéré de n tâches est donc

$$\sum_{i \in \mathcal{J}} w_i C_i = \sum_{i \in \mathcal{J}} w_i (H + 1 - \sum_{k \in \mathcal{T}} c_{i,k}) = \sum_{i \in \mathcal{J}} w_i (H + 1) - \sum_{i \in \mathcal{J}} \sum_{k \in \mathcal{T}} w_i c_{i,k} \tag{A.2}$$

Ainsi, pour le premier modèle, nous présentons les variables suivantes :

- $x_{i,k}$ l'état de la tâche $i \in \mathcal{J}$ à l'heure $k \in \mathcal{T}$
- $u_{i,k}$ variable de décision de la quantité de ressources utilisée par la tâche $i \in \mathcal{J}$ au moment $k \in \mathcal{T}$
- $c_{i,k}$ variables d'indication d'achèvement d'une tâche. Il est égal à 1 si job $i \in \mathcal{J}$ a déjà été complété à l'instant $k \in \mathcal{T}$. Sinon, il est 0.
- $y_{i,k}$ processing state indication variables, 1 si job $i \in \mathcal{J}$ est en cours de traitement $k \in \mathcal{T}$, 0 sinon.

Contraintes de disponibilité temporelle (Fenêtre de temps)

$$y_{i,k} = 0 \quad \forall i \in \mathcal{J}, k \in \mathcal{T} \setminus \{r_i, ..., d_i\} \tag{A.3}$$

$$y_{i,k} \in \{0, 1\} \quad \forall i \in \mathcal{J}, k \in \{r_i, ..., d_i\} \tag{A.4}$$

Consommation de ressource semi-continue Lorsqu'une tâche est traitée à l'instant k, la ressource qui lui a été attribuée doit être inférieure à $\bar{u}_i$ et supérieure à $\underline{u}_i$.

$$u_{i,k} \leq \bar{u}_i . y_{i,k} \quad \forall i \in \mathcal{J}, k \in \{r_i, ..., d_i\} \tag{A.5}$$

$$u_{i,k} \geq \underline{u}_i . y_{i,k} \quad \forall i \in \mathcal{J}, k \in \{r_i, ..., d_i\} \tag{A.6}$$

Non-préemption

$$y_{i,k} \geq y_{i,k-1} - c_{i,k} \tag{A.7}$$

$$y_{i,k} \leq 1 - c_{i,k} \tag{A.8}$$

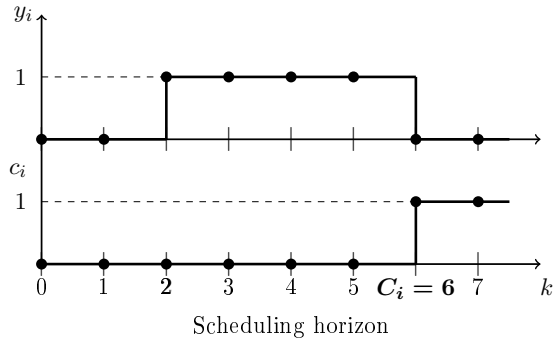


FIGURE A-4 – Variables de décision $y_{i,k}$ et $c_{i,k}$

État de complétion

$$x_i^* c_{i,k} \leq x_{i,k} \quad \forall i \in \mathcal{J}, k \in \mathcal{T} \tag{A.9}$$

$$c_{i,k} = 0 \quad \forall i \in \mathcal{J}, k \in \mathcal{T} : k < e_i \tag{A.10}$$

$$c_{i,k} \in \{0,1\} \quad \forall i \in \mathcal{J}, k \in \mathcal{T} : k \geq e_i \tag{A.11}$$

Dans l'équation (A.9), l'indicateur d'achèvement peut être écrit comme suit : $c_{i,k} \leq \frac{x_{i,k}}{x_i^*}$. Ensuite, il est possible de passer à «1» uniquement si l'état de la tâche i est supérieur à la charge de travail de cette tâche. En outre, l'indicateur d'achèvement est obligé d'être nul avant le premier temps d'achèvement.

Formulation cumulative : P-Cml

$$\textit{Minimiser} \quad \sum_{i \in \mathcal{J}} w_i(H+1) - \sum_{i \in \mathcal{J}} \sum_{k \in \mathcal{T}} w_i c_{i,k} \tag{A.12}$$

$$s.t. \quad (3.4 - 3.6)$$

$$(A.3 - A.11)$$

A.3.4 Formulation disjonctive

En ce qui concerne la formulation disjonctive, nous modélisons l'état de traitement et l'achèvement du travail directement par des variables binaires disjonctives. Ces variables disjonctives appartiennent au SOS1 comme nous l'avons mentionné au début du chapitre.

- $\alpha_{i,k}$ état de commencement, 1 si tâche $i \in \mathcal{J}$ commence à traiter à l'heure $k \in \mathcal{T}$, 0 sinon.

- $\beta_{i,k}$ état de complétion, 1 si tâche $i \in \mathcal{J}$ se termine à l'heure $k \in \mathcal{T}$, 0 sinon.

Donc

$$C_i = \sum_{k \in \mathcal{T}} (k\beta_{i,k}) \quad \forall i \in \mathcal{J} \tag{A.13}$$

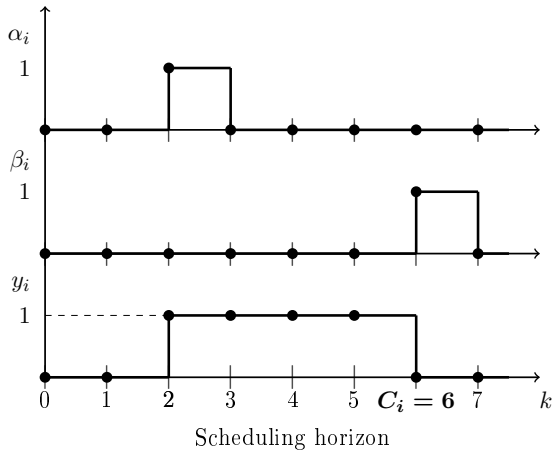


FIGURE A-5 – Variables de décision $\alpha_{i,k}$ et $\beta_{i,k}$

Fenêtre de temps

$$\alpha_{i,k} = 0 \quad \forall i \in \mathcal{J}, k \in \mathcal{T} \setminus \{r_i, ..., l_i\} \tag{A.14}$$

$$\alpha_{i,k} \in \{0,1\} \quad \forall i \in \mathcal{J}, k \in \{r_i, ..., l_i\} \tag{A.15}$$

$$\beta_{i,k} = 0 \quad \forall i \in \mathcal{J}, k \in \mathcal{T} \setminus \{e_i, ..., d_i\} \tag{A.16}$$

$$\beta_{i,k} \in \{0,1\} \quad \forall i \in \mathcal{J}, k \in \{e_i, ..., d_i\} \tag{A.17}$$

Avec les formulations de l'ensemble des contraintes, toutes les borne e_i et l_i sont utilisées dans $P{-}Dsj$ tandis que l'ensemble équivalent de contraintes dans le modèle $P{-}Cml$ peut uniquement utiliser un e_i.

Non-préemption

$$\sum_{k=r_i}^{l_i} \alpha_{i,k} = 1 \quad \forall i \in \mathcal{J} \tag{A.18}$$

$$\sum_{k=e_i}^{d_i} \beta_{i,k} = 1 \quad \forall i \in \mathcal{J} \tag{A.19}$$

$$\tag{A.20}$$

Avec ces contraintes, on peut trouver qu'il y a toujours une seule possibilité pour qu'un élément de α ou β soit un. Par conséquent, la formulation s'appelle *disjonctive*.

Complétion

$$\beta_{i,k}.\tilde{x}_i \leq x_{i,k} \quad \forall i \in \mathcal{J}, k \in \mathcal{T} \tag{A.21}$$

Formulation disjonctive : P-Dsj

$$\begin{aligned} \textit{Minimize} \quad & \sum_{i \in \mathcal{J}} \sum_{k \in \mathcal{T}} (w_i.k.\beta_{i,k}) \\ s.t. \quad & (3.4 - 3.6) \\ & (A.14 - A.21) \end{aligned} \tag{A.22}$$

Nous introduisons deux coupes valides pour renforcer la formulation

Coupe 1 $\beta_{i,k} \leq \sum_{\tau=e_i}^{k} \alpha_{i,\tau},\ i \in \mathcal{J},\ k \in \{e_i, ..., d_i\}$ est valide.

Coupe 2 $\beta_{i,k} + \alpha_{i,k} \leq 1\ i \in \mathcal{J},\ k \in \{r_i, ..., d_i\}$ est valide

Preuves Voir chapitre 3

A.3.5 Analyse convexe

On peut voir que la différence entre les deux formulations est marquée par les contraintes (A.7), (A.8) vs (A.18), (A.19) et (3.29). Prenons un exemple pour dessiner la région faisable des problèmes de relaxation LP liés à ces contraintes. Nous considérons un problème de planification avec $H = 2$ et $n = 1$. Les figures 3-8 et 3-9 montrent la région faisable des variables y, c et α, β sur le LP-Relaxation des $P - Cml$ et $P - Dsj$. Elles montrent que la formulation $P - Dsj$ est plus stricte que celle de la formulation $P - Cml$.

A.3.6 Test numériques

Étant donné qu'il n'y a pas de problème existant qui traite en même temps le profil de puissance dépendant du temps et les contraintes temps-fenêtre alors nous introduisons un générateur d'instance aléatoire adhoc. La création des instances d'entrée aléatoire est décrite dans la figure 3-13.

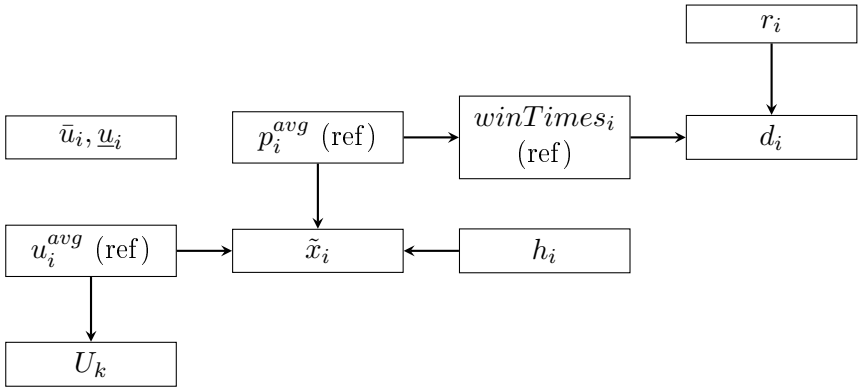


FIGURE A-6 – Générateur aléatoire

Il existe cinq modèles différents qui contiennent deux formulations principales $P{-}Cml$ et $P - Djs$ avec tous les coupes incluses. En outre, il existe trois variantes de $P - Dsj$: la formulation disjointe avec Coupe 1 (Coupe (3.29)), avec Coupe 2 (Coupe (3.34)) et sans aucune coupe A noté $P - Djs - Cut1$, $P - Djs - Cut2$ et $P - Djs - noCut$.

Pour chaque test, nous enregistrons le temps d'exécution ($Heure$), le nombre de nœuds résolus ($\#Nds$) et PLMNE -Gap ($MIPGap$). Comme nous l'avons mentionné plus tôt dans le chapitre, le solveur utilise la branche et l'algorithme lié pour résoudre le modèle.

L'écart PLMNE est calculé par la fraction de la différence entre la limite supérieure du titulaire et la borne inférieure du titulaire avec la limite supérieure : $MIPGap = \frac{UB-LB}{UB}$. L'intervalle de temps de chaque modèle PLMNE est obtenu par : $BTGap_i = \frac{T_i - \min_i T_i}{T_i}$. Les résultat des tests numérique sont dans les tableaux 3.4 et 3.5. Le résultat des tests numériques confirme notre observation sur l'analyse de la relaxation LP où $P-Dsj$ dépasse la performance de la formulation $P-Cml$. Cette formulation se prouve son efficacité faisant face du problème.

Malgré tout, nous cherchons une méthode de résolution plus efficace et dédié au problème ACPV 2. C'est pour cette raison que nous introduisons l'algorithme de type Branch-and-Price dans la section suivante.

A.4 Résolution exacte : Algorithme Branch-and-Price

A.4.1 Contexte

L'ACPV 2 nécessite un nombre important de variables de décision lorsque la taille du problème augmente drastiquement, en particulier dans la dimension de l'horizon de planification. Ainsi, la méthode de Branch-and-Price, qui a prouvé beaucoup de succès dans la résolution d'énormes problèmes PLMNE [8], est un choix approprié pour aborder notre problème. En outre, le problème ACPV 2 a un caractère décomposable. La plupart des contraintes (fenêtre de temps, demande de ressources, consommations de ressources ...) se concentre sur le problème local, ce qui signifie que ces contraintes ne se limitent qu'à des tâches localement. Il n'y a qu'une contrainte globale, la disponibilité des ressources, qui regroupe tous les problèmes locaux.

A.4.2 Problème maître

Pour construire le problème maitre, tout d'abord nous introduisons le sous-problème local de chaque tâche :

Formulation locale du sous-problème pour la rangée i *LCi*

$$J_{1,l_i}\alpha_i^T = 1 \tag{A.23}$$

$$J_{1,l_i}\beta_i^T = 1 \tag{A.24}$$

$$\beta_i^T \leq \Delta_{l_i}\alpha_i^T \tag{A.25}$$

$$J_{1,l_i}u_i^T \geq \tilde{x}_i \tag{A.26}$$

$$u_i \geq \underline{u}_i\Delta_{l_i}(\alpha_i - \beta_i)^T \tag{A.27}$$

$$u_i \leq \bar{u}_i J_{1,l_i} \tag{A.28}$$

$$\alpha_i, \beta_i \in \{0,1\}^{l_i} \tag{A.29}$$

Les matrices de coefficients sont définies comme suit :

- $J_{n,m}$ est une matrice unité de taille $n \times m$.

- Δ_{l_i} est une matrice triangulaire inférieure avec une taille $l_i \times l_i$: $\Delta_{i,j} = 1$ si $i \leq j$, $= 0$ *sinon*.

Chaque point extrême de la région faisable du problème LCi constitue une colonne. Soit Ω_i l'ensemble de tous les points extrêmes de la région réalisable créé par LCi. $x_i^\omega = (u_i^\omega, \alpha_i^\omega, \beta_i^\omega)$ soit un point extrême de Ω_i. Le coût de cette allocation au problème d'origine peut être exprimé en $y_i^\omega = f(C_i^\omega) + f(r_i)$ avec $C_i^\omega = \sum_{k=0}^{l_i-1} w_i k \beta_{i,k}^\omega$ Par conséquent, LCi peut être exprimé comme une combinaison convexe :

$$u_i = \sum_{\omega \in \Omega_i} \lambda_{i\omega} u_i^\omega \tag{A.30}$$

$$\alpha_i = \sum_{\omega \in \Omega_i} \lambda_{i\omega} \alpha_i^\omega \tag{A.31}$$

$$\beta_i = \sum_{\omega \in \Omega_i} \lambda_{i\omega} \beta_i^\omega \tag{A.32}$$

$$\sum_{\omega \in \Omega_i} \lambda_{i\omega} = 1 \tag{A.33}$$

$$\lambda_{i\omega} \geq 0 \; \omega \in \Omega_i \tag{A.34}$$

$$\alpha_i, \beta_i \in \{0,1\}^{l_i} \tag{A.35}$$

Avec coût $y_i = \sum_{\omega \in \Omega_i} \lambda_{i\omega} y_i^\omega$.

De là, on peut déduire la relaxation LP du problème maitre :

$$\text{Minimiser} \qquad z = \sum_{i \in \mathcal{J}} \sum_{\omega \in \Omega_i} \lambda_{i\omega} y_i^\omega \tag{A.36}$$

$$\text{Sous constraintes de} \qquad \sum_{i \in \mathcal{J}} \sum_{\omega \in \Omega_i} \lambda_{i\omega} u_i^\omega \leq R \tag{A.37}$$

$$\sum_{\omega \in \Omega_i} \lambda_{i\omega} = 1 \qquad i \in \mathcal{J} \tag{A.38}$$

$$\lambda_i \geq 0 \qquad i \in \mathcal{J} \tag{A.39}$$

A.4.3 Problème de pricing

Soit (π, ϕ) la paire de vecteurs de variables doubles correspondant respectivement aux contraintes de disponibilité de ressources totales (ou aux contraintes de capacité du système) (A.37) et aux contraintes de convexité (A.38) de MLP , Donc $\pi \in \mathbf{R}_{-}^{H}$ et $\phi \in \mathbf{R}_{+}^{n}$. Le double programme de MLP, noté DLP, est explicitement déduit de la formulation MLP comme suit.

$$\text{Maximiser} \qquad z_D = R\pi + \sum_{i \in \mathcal{J}} \phi_i \tag{A.40}$$

$$\text{Sous constraintes de} \qquad u_i^{\omega}\pi + \phi_i \leq y_{\omega i},\ i \in \mathcal{J},\ \omega \in \Omega_i \tag{A.41}$$

$$\pi_k \leq 0,\ k \in \mathcal{K}$$

$$\phi_i > 0,\ i \in \mathcal{J}$$

Comme il est impossible d'énumérer explicitement toutes les colonnes dans $\Omega_i, \quad i = 1, \ldots, n$ en raison de sa taille exponentielle, nous définissons un ensemble restreint de Ω_i qui est noté Ω_i'. L'ensemble restreint est ainsi créé avec une taille raisonnable pour ne contenir que des colonnes significatives qui améliorent la valeur objective du problème principal. Le problème restreint est élargi en ajoutant plus de colonnes à la phase de pricing. Le problème maître qui ne contient que l'ensemble des colonnes Ω_i' est appelé le problème maître restreint MLP(Ω').

Pour ajouter les nouvelles colonnes, il faut résoudre le problème de pricing avec les solutions duals actuelles π^* et ϕ_i^* de MLP(Ω').

$$\text{Minimiser} \qquad c_i^* = y_{\omega i} - u_i^{\omega}\pi^* - \phi_i^* \tag{A.42}$$

$$\text{Sous constraintes de} \qquad \text{LCi}$$

A.4.4 Stratégies de branchement

Au nœud de départ, on initie une solution possible au problème de RMP en utilisant une heuristique. La relaxation de ce RMP est résolue, puis le double de la relaxation LP est utilisé pour évaluer tous les sous-problèmes. Si la (les) colonne(s) avec coût(s) négatif(s) est(sont) trouvé(s), le ou les coûts les plus négatifs seront choisis pour entrer en fonction. Sinon, si l'optimum de relaxation RMP est intégral, il est également optimal pour le MP et l'algorithme s'arrête. Dans le dernier cas, si la solution de relaxation n'est pas intégrale, et aucune colonne ne peut être ajoutée, il faut brancher l'arbre pour de nouveaux nœuds. La génération de la colonne itère de nouveau à chaque nouveau nœud.

Branchement sur la flexibilité de l'ensemble SOS1 Étant donné que les fenêtres de temps sont contraintes par les variables binaires α et β, le schéma de dérivation est proposé de manière à ce que les fenêtres de temps du travail les plus flexible (MFlex) / les moins flexibles (LFlex) soient dichotomisées pour former un arbre binaire. Une illustration est donnée en figure A-7 pour la stratégie *MFlex*.

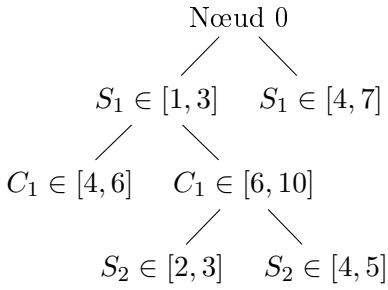


FIGURE A-7 – Un exemple de branchement

Algorithm 12: Branchement sur la flexibilité de l'ensemble SOS1
if *Pouvoir trouver le plus (moins) intervalle flexible* $S_i^* \in [s_a, s_b]$ *(ou* $C_i^* \in [C_a, C_b]$*) avec* $s_b - s_a > 0$ **then**
  $\kappa = min\{t : \sum_{k=s_a}^{t} \alpha_k \geq 1/2\}$;
  Ou $\kappa = min\{t : \sum_{k=C_a}^{t} \beta_k \geq 1/2\}$;
  Créer nœud $S_i^1 \in [s_a, \kappa]$ et $S_i^2 \in [\kappa + 1, s_b]$;
  Ou $C_i^1 \in [C_a, \kappa]$ et $C_i^2 \in [\kappa + 1, C_b]$;
else
  Plus de nœud enfant à créer
end

A chaque nœud, après la phase de génération de colonne, une décision sera prise selon

les règles standard de la séparation et évaluation.

Sélection de nœud Nous proposons deux stratégies de recherche d'arbres : La meilleure borne LP en premier (Best LP Bound first, BLF) et La meilleure projection en premier (Best Projection first, BPF) [1]. Dans la première, le nouveau nœud sélectionné pour créer l'enfant est celui qui a la plus petite borne LP. Dans le cas de l'égalité, celui avec un niveau plus profond a priorité (Profondeur-Première orientation). Dans cette approche, la projection d'un nœud est calculée comme suit :

- Somme des fractionnements $s^i = \sum_j min(f_j, 1 - f_j)$ pour f_j la valeur fractionnaire de la variable binaire actuelle. La somme des fractionnements du nœud parent est notée s^0.
- La borne LP du problème maître restreint sur le nœud z_u^i
- Le Global (meilleur) LP lié z_L
- La meilleure intégrale actuelle intégrée z_{IP}

Par conséquent, la valeur de projection estimée au nœud i est définie par $E_i = z_u^i + (z_{IP} - z_L)\frac{s^i}{s^0}$. La meilleure projection estimée sélectionne le nœud à $argmin\{E_i\}$.

A.4.5 Bornes duales pour stabiliser la génération de colonne

Pour resserrer la limite inférieure de chaque nœud, nous appliquons la borne duale [74] pour une terminaison anticipée de l'itération de la génération de colonne pour éviter l'effet de rétention, et pour un élagage précoce dans l'arbre dérivé et relié .

$$LB_{RMP} = z^*_{RMP} + \sum_{i^* \in \mathcal{J}^*} c^*_{\omega i^*} \tag{A.43}$$

Où z^*_{RMP} est la valeur réelle LP-détendue de l'objectif du problème maître restreint, $c^*_{\omega i^*}$ est le coût réduit des nouvelles colonnes entrant dans la base pendant la phase de prix où $\mathcal{J}^*$ est l'ensemble des sous-problèmes donnant un coût réduit négatif. Cette limite évite d'ajouter

des colonnes qui n'améliorent pas la relance liée par la procédure de génération de la colonne d'arrêt lorsque $\lceil LB_{RMP} \rceil = \lceil LB \rceil$. Dans le cas où la génération de colonne est exécutée sur un nœud, si $\lceil LB_{RMP} \rceil \geq Incumbent$, alors il peut être élagué.

A.5 Interprétation des résultats des tests numériques

Les détails sur la configuration des tests peuvent être trouvés au chapitre 4. Les résultats numériques sont visible dans tableau 4.1 et tableau 4.2.

Dans la plupart des cas testés, la configuration de branchement *MFlex* surpasse d'autres configurations. Les premières solutions intégrales peuvent être trouvées rapidement par l'algorithme de type Branch-and-Price (la solution trouvée par heuristique n'est pas prise en compte). L'effet de mise au rebut est beaucoup plus important pour les cas exceptionnels, où l'optimalité prend beaucoup de temps pour être prouvée. Au cours de la surveillance des tests numériques, dans la plupart des grands exemples, on peut remarquer qu'il faudrait un temps relativement court pour trouver des solutions quasi optimales. Il est rapide de trouver des écarts $< 1\%$, même l'écart $< 0,3\%$ dans la majorité des cas sur les deux approches Best-LP-First et Best-Projection-First. Les résultats numériques soulignent une grande sensibilité de l'algorithme à la stratégie de branchement. Notre meilleure stratégie de projection offre une excellente performance lors des tests.

A.6 Résolution approximative : Heuristique HLA

L'**ACPV 2** est NP difficile au sens fort, il est alors difficile de trouver la solution exacte pour l'instance de grande taille dans un délai raisonnable. En outre, la problématique industrielle nécessite une réponse rapide pour l'instance de la vie réelle. Une heuristique bien construite pourrait résoudre ce problème en trouvant une bonne solution dans un délai raisonnable.

A.6.1 Procédure globale

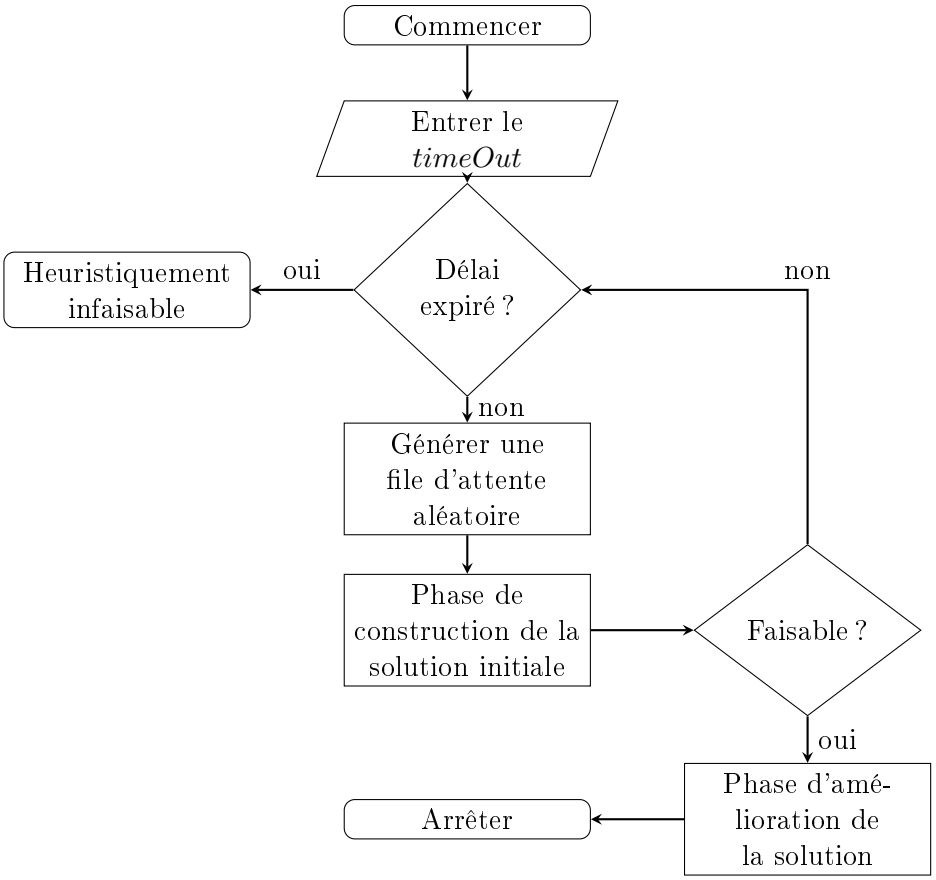


FIGURE A-8 – Heuristique HLA

A.6.2 Phase de construction de la solution initiale : Superposition de la ressource

On utilise une heuristique qui applique les coupes de type guillotine et le principe de mise en couche d'allocation d'une ressource pour construire la solution initiale. L'idée de l'algorithme est de planifier sans créer de nouveau pic de ressource, ce qui signifie que la consommation totale maximale de ressources se situe sur l'horizon de planification. Dans le cas où nous nous obligeons à créer un nouveau pic de ressources, le placement du travail devrait être l'endroit où le pic nouvellement créé est minimal. À chaque coupure de guillotine provisoire, on essaie de guillotiner l'élément d'emballage (c'est-à-dire l'allocation de ressources pour un travail donné) afin de réduire la hauteur réelle de ressource. Les composants algorithmiques sont présentés dans 3, 4 et 6.

A.6.3 Phase d'amélioration de la solution

Cette section propose trois méthodes de recherche à grande distance pour améliorer la solution initiale.

Occupation gourmande L'objectif du processus de la superposition de la ressource est d'aplanir le pic de l'allocation. Par conséquent, cela laisse beaucoup de ressources disponibles. L'heuristique de l'occupation gourmande tente de tirer le plus possible la ressource laissée pour attribuer aux tâches. Le processus d'occupation gourmand est détallé comme suit. L'heuristique analyse chaque intervalle de temps, s'il y a une quantité disponible de ressources non allouées, alors l'heuristique va distribuer avec avidité aux tâches. L'occupation gourmande est basée sur le principe de recherche de grand voisinage. Bien qu'il y ait une ressource non consommée à l'heure k, les tâches avec moins de $\tilde{x}_{i^*}$ et a déjà commencé à traiter au moment k seront supprimées de la bande d'alimentation. La tâche i^* sera reconditionnée avidement dans la bande par l'algorithme 7. La construction du travail i^* par l'algorithme 7 conduit toujours à une solution réalisable, car le pire des cas est déjà l'allocation des ressources de la tâche précédente avant le retrait.

Tremblement de forme libre On fixe un nombre de paires de tâches aléatoire, noté $nPairs$. Pour chaque paire, les tâches seront supprimées de la bande d'alimentation et on tente alors de réintroduire la bande de puissance par algorithme 7. L'algorithme est détaillé dans **??**. Toutes les meilleures solutions trouvées à pendant chaque itération de tremblement devient la meilleure solution.

Tremblement des rectangles Le tremblement des rectangles a le même mécanisme de destruction que l'algorithme de tremblement de forme libre. Cependant, la reconstruction des tâches est une allocation de ressources moulables, c'est-à-dire que les tâches doivent consommer des ressources constantes.

A.6.4 Interprétation des résultats des tests numériques

Les détails sur la configuration des tests peuvent être trouvés au chapitre 5. Les résultats numériques sont trouvés dans les tableaux 5.4 et 5.5.

L'écart entre la solution trouvé par les deux heuristiques HLA et la solution optimale diminue sur la taille du problème dans le terme de nH^2. Selon l'écart moyen trouvé, on peut conclure que les HLA heuristiques se comportent bien avec les deux séries de tests. Plus précisément, on peut constater que le HLA-RAG montre des solutions légèrement meilleures dans les cas meilleurs et moyens, tandis que HLA-ARG maintient le meilleur *Gap*. HLA-RAG maintient également une marge plus stricte entre le meilleur cas et le pire des cas, donc il est plus stable en terme de qualification de solution.

Dans les instances les plus large, $n = 200$ et $H = 200$, HLA-RAG prend moins de deux minutes pour trouver une solution finale tandis que HLA-ARG prend moins de 2,7 minutes. Le temps d'exécution des deux techniques pour trouver des solutions finales est rapide par rapport à la taille de l'instance à traiter. L'écart entre le meilleur et le pire moment trouvé pour chaque instance est relativement faible, donc les deux versions HLA sont stables en terme de temps d'exécution. Pour plus de détails, HLA-RAG surpasse HLA-ARG en ayant en moyenne des temps d'exécution plus courts pour les deux ensembles testés. Pour une petite instance, la différence entre le meilleur et le pire écart est relativement importante,

mais le temps d'exécution est vraiment petit.

Dans le cas industriel actuel, l'heuristique a prouvé une performance raisonnablement bonne. Plus précisément, selon les tests de mise en œuvre de *HLA-RAG* traitant de nos derniers problèmes industriels contenant jusqu'à 25 tâches dans un schéma de planification de 12 heures, chaque intervalle dure 15 minutes (48 intervalles au total), la dérivation moyenne trouvée entre les solutions obtenues par *HLA-RAG* et la solution optimale résultant de CPLEX est de 1.7% tandis que le délai d'exécution de l'heuristique est réglé sur 20 secondes.

A.7 Prévisions basées sur la simulation et ordonnancement en temps réel

A.7.1 Modèle prévisionnel

Nous présentons le protocole de simulation dans deux aspects : la variation de la largeur de bande de puissance et la variation de la longueur d'ordonnancement. Les résultats statistiques des deux simulateurs sont utilisés pour définir une largeur de bande de puissance optimale et un horizon de planification standard. En outre, pendant l'opération, une fois que les paramètres du générateur aléatoire ont changé selon la nouvelle tendance des données de la vie réelle, la base de référence du calendrier peut être construite pour guider la gestion en temps réel. Par exemple, si la ligne de base indique que la largeur de bande de puissance actuelle n'est pas suffisante pour que tout VE soit complètement chargé dans l'horizon de temps donné, toute l'opération peut être démarrée plus tôt.

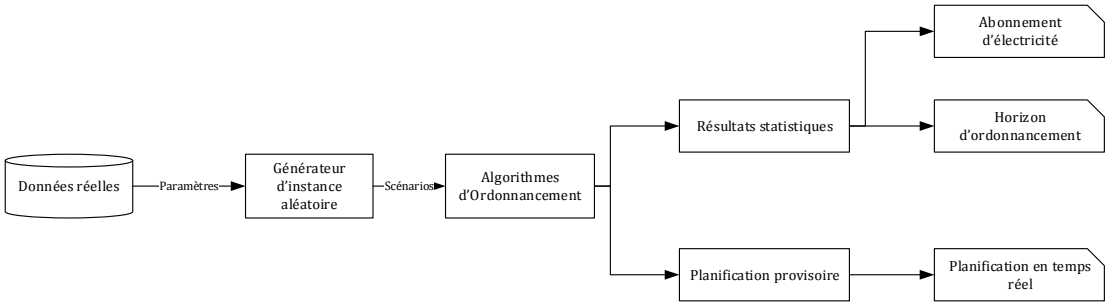


FIGURE A-9 – Prévisions basées sur la simulation

Étude de cas L'étude de cas prend en entrée des données de notre projet industriel de la gestion de facturation pour le stationnement résidentiel en France. Une base de données historique de 30 VE est reçue pendant la période de mars 2016 à mai 2016. Les temps d'arrivée sont aléatoires, mais le départ de chaque véhicule est fixé à 6 heures par jour par défaut. Selon les données recueillies, nous choisissons la distribution gamma pour simuler la consommation quotidienne et la distribution normale pour simuler les temps d'arrivée. Les histogrammes sur les données réelles collectées (figure 6-8) justifient nos choix de distributions aléatoires pour chaque type de données simulées.

On lance le simulateur d'horizon de planification fixe (Fig. A-10). Le résultat du

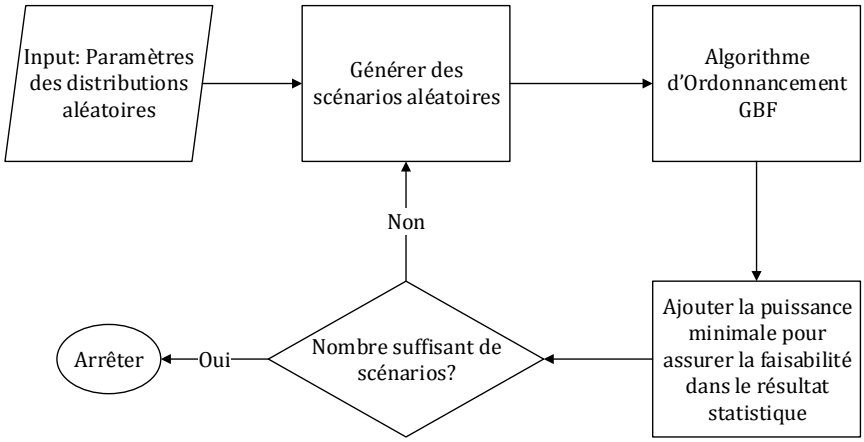


FIGURE A-10 – Description du simulateur d'horizon temporel fixe

simulateur d'horizon à temps fixe se trouve sur les figures A-11 et A-12. La figure A-11 montre la densité des occurrences de puissance de charge totale maximale sur plus de 10000 instances aléatoires.

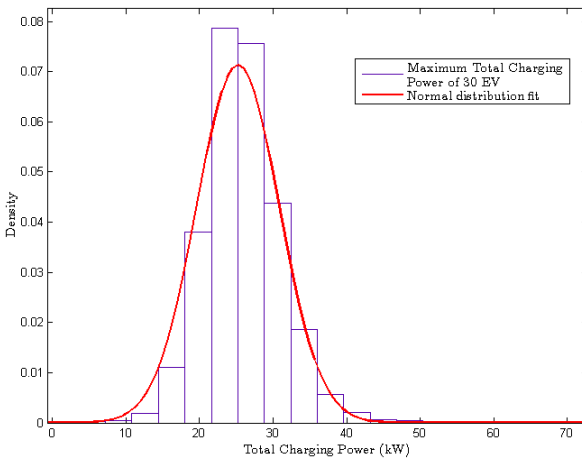


FIGURE A-11 – L'histogramme de la puissance de charge totale maximale de 30 VE résulte de 10000 tests

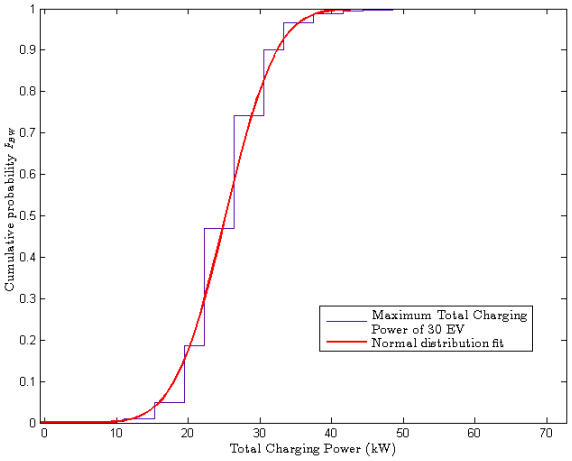


FIGURE A-12 – La fonction cumulative de probabilité correspondante F_{BW} et sa répartition correspond à l'équivalent

On lance le simulateur de bande passante fixe (Fig.A-13)

La figure A-12 indique que la fonction cumulative de distribution F_{BW} et sa ré-

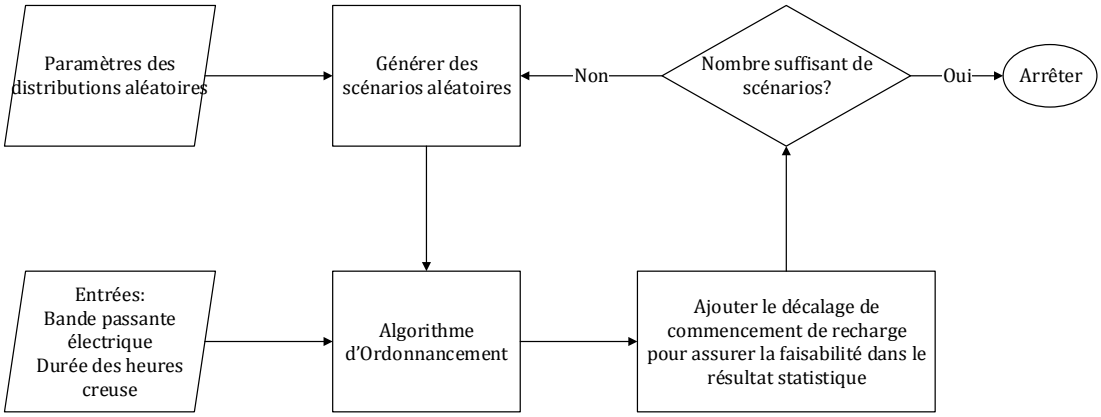


FIGURE A-13 – Description du simulateur de bande passante fixe

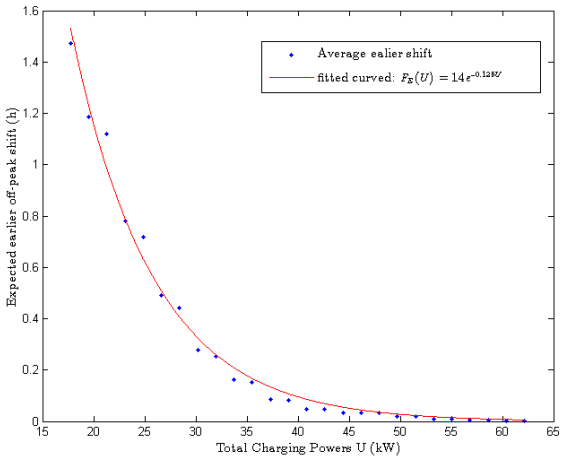


FIGURE A-14 – Décalage prévu (h) pour commencer avant les heures creuses pour assurer la faisabilité en fonction de la bande passante utilisée ($n_{VE} = 30$ et $n_{tests} = 10000$)

partition correspondent à la distribution. Selon l'histogramme, nous pouvons estimer que l'utilisation de l'énergie totale suit une distribution normale dont la moyenne est de 16,7 kW (environ 14,2 % de la somme de toute la puissance de charge, c'est-à-dire **foisonnement** électrique) et sa écart type est de 5,6 kW.

Les résultats statistiques ont aussi servi à chercher une valeur optimale de l'abonnement électrique afin de minimiser la somme du coût fixe et du coût variable de l'utilisation d'électricité.

A.7.2 Ordonnancement en temps réel

Étant donné que les tâches ne sont pas préemptives, la réparation de l'horaire devrait être partielle. Seuls les travaux dans la file d'attente sont sur le point d'être reportés, les tâches qui ont déjà commencé continuent à être rechargées et constituent une contrainte pour la mise à jour du programme. Le planificateur en ligne commence au "temps zéro" décidé par les prévisions basées sur la simulation. Ensuite, il planifie les emplois déjà disponibles dans le système (VE branché). Le point de charge exécute l'affectation des ressources selon la planification provisoire et attend des nouveaux événements. Pour chaque nouvel événement, le planificateur modifie l'ensemble J' des emplois disponibles dans le système et l'ensemble Ω d'exécution des travaux. Selon la politique de ré-ordonnancement partiel, l'algorithme déterministe planifie alors la nouvelle entrée J' Ω. Un nouvel horaire pour les travaux dans la file d'attente est établi et le planificateur en ligne l'exécute.

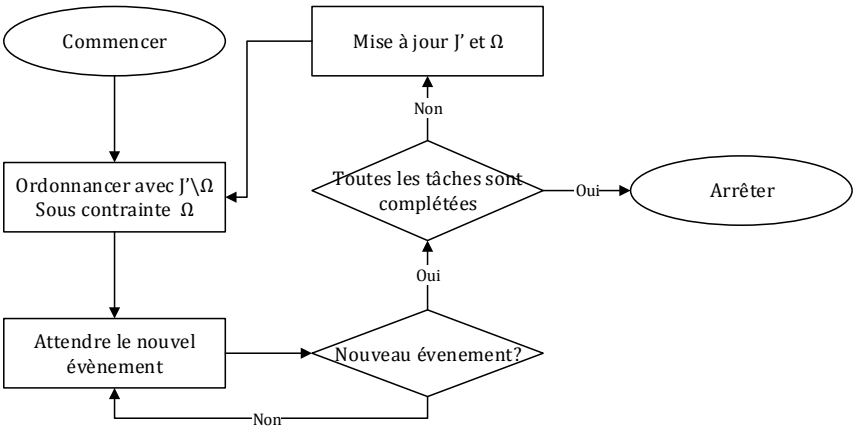


FIGURE A-15 – La politique d'ordonnancement en temps réel

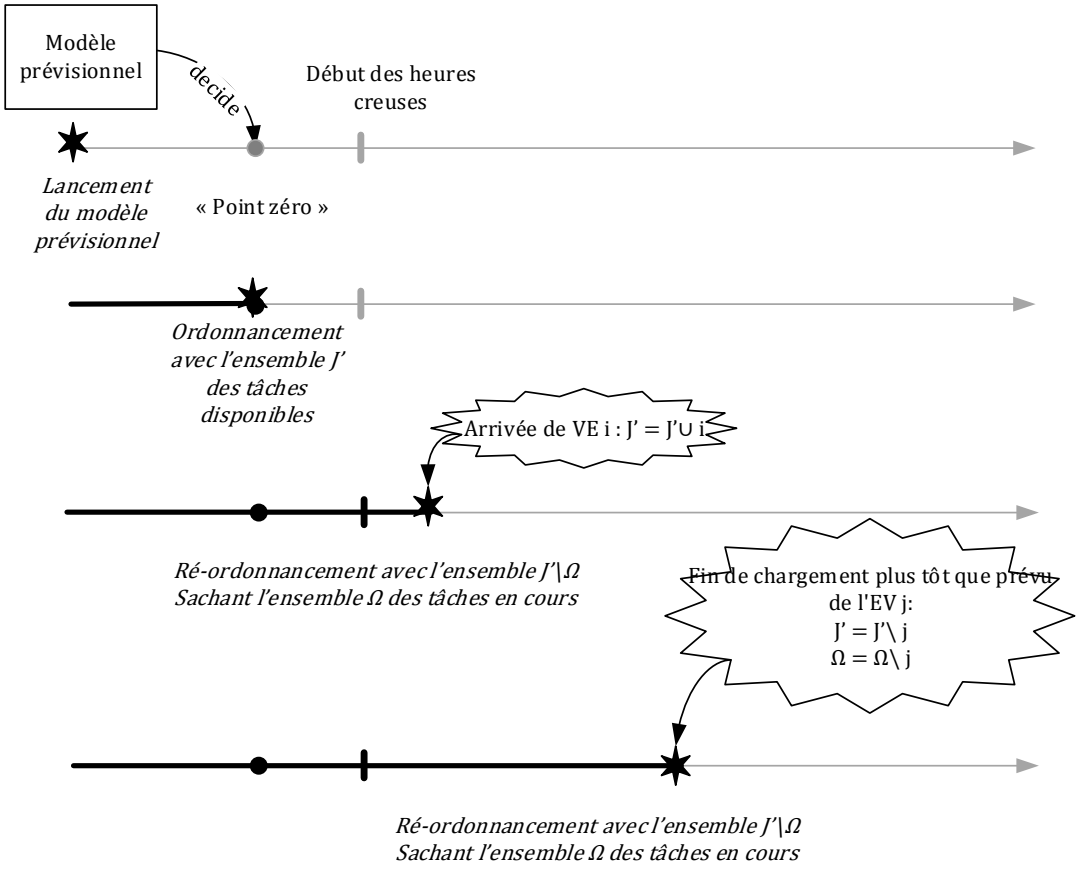


FIGURE A-16 – La politique de l'ordonnancement en temps réel dans l'exemple

A.8 Le logiciel

Gestion opérationnelle (GO) Le logiciel de gestion du box NEMO

Platforme algorithmique (PA) Plate-forme sur laquelle la boîte à algorithmes et la gestion opérationnelle sont implémentées. Elle garantit également une bonne communication inter-procédures et gère la sécurité.

Boîte d'algorithmes (BA) La boîte contient des algorithmes de procédure de solution. Il rend automatiquement le choix de la méthode de calcul après chaque demande de gestion opérationnelle. Il existe deux composantes :

- Bibliothèque d'algorithmes d'ordonnancement (BIB-ALG). L'ensemble des méthodes mises en œuvre pour répondre aux différentes situations actuelles. Version actuelle : BIB-ALG 1.0
- Le module de prévision basé sur la simulation (MOD-PREV). Le module traite la base de données NEMO (NemoDB) vers la base de données historique (HistDB).

Unité de gestion de la communication (CMU) Le module par lequel la boîte NEMO peut communiquer avec le BA par un ensemble d'événements prédéfini et traduit le scénario reçu par BA pour le contrôle des terminaux NEMO.

La figure A-17 détaille les flux d'informations entre les composants.

Le PA est créé pour assurer en même temps un système de ordonnancement flexible avec une synchronisation étroite des composants électroniques gérés par le GO. La synchronisation entre GO et BA est pilotée par la CMU, détaillée en figure A-18. Pour chaque événement inattendu capturé, si le BA n'est pas occupé, il demande la nouvelle mise à jour sur le calendrier. Sinon, il empile toutes les modifications, puis envoie de nouvelles informations dès que le BA sera disponible.

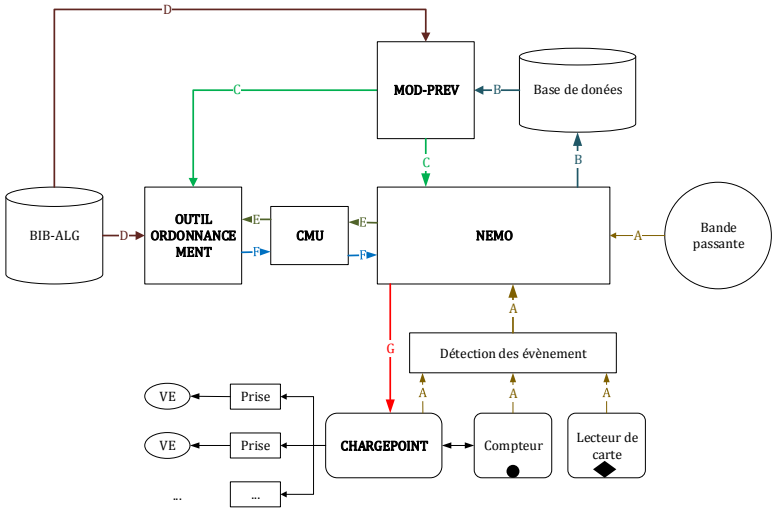


FIGURE A-17 – Les flux d'informations

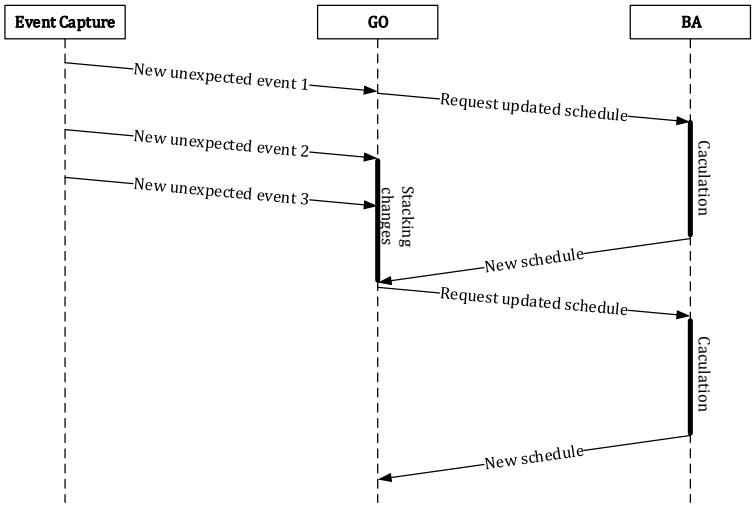


FIGURE A-18 – Le CMU

A.9 Conclusion et Perspectives

Dans le contexte du développement durable et de l'utilisation de l'énergie propre, la recherche sur les technologies des véhicules électriques a attiré beaucoup d'attention. Plus que jamais, nous faisons face à une croissance rapide de la vente de véhicules électriques. Par conséquent, la recherche sur la gestion de la tarification de l'EV revêt une grande importance. Par la même raison, l'optimisation de la coordination de l'EV est en constante évolution. Au milieu de la vague actuelle de recherche mondiale, cette thèse tend à proposer une solution mesurée pour le problème de coordination de la charge EV dans le contexte spécifique de la France. Le travail effectué dans cette thèse est le fruit d'une convention industrielle avec le partenariat de la société **Park'nPlug** basée sur Rosières-près-Troyes, en France et le **Laboratoire d'Optimisation des Systèmes Industrielles** de **UTT**, France. Pour être pionnier dans la nouvelle technologie, une recherche d'une « Recharge intelligente » peut assurer un calendrier optimal et une fonctionnalité autonome qui sont également les objectifs de la recherche de cette thèse.

Tout d'abord, l'état de l'art a souligné l'écart de recherche sur un problème de planification, où les tâches sont malléables (ou semi-malléables) en tenant compte d'un profil de ressources variable dans le temps. Par cette analyse, nous pouvons positionner notre recherche dans cette thèse pour combler l'écart spécifique. La méthodologie était appropriée, ce qui peut nous fournir une base solide pour tous les développements. La création de nouvelle classe de problèmes d'ordonnancement, avec l'identification des contraintes CRVE, s'attaque directement au problème de la vie réelle. En outre, cinq configurations : ACPF 1/2 et ACPV 1a / 1b / 2 nous aident à classer le problème industriel existant. L'analyse de complexité de chaque problème selon l'objectif d'optimisation standard suggère la méthode de résolution appropriée correspondant à chaque configuration.

Avec une approche scientifique, nous avons analysé différentes techniques pour formuler le ACPV 2, le problème le plus générique et le plus difficile à résoudre parmi les cinq configurations. Les deux techniques choisies l'ont été pour formuler le problème de manière conjonctive ($P-Cml$) et disjonctive ($P-Dsj$). Une analyse complète de l'enveloppe convexe et de la relaxation LP conduit à l'introduction des deux coupes pour les $P-Dsj$. Les tests numériques ont vérifié notre analyse théorique sur les deux formulations et ont déclaré que

la formulation la plus forte est le $P - Dsj$ avec une seule coupe 1.

Puisque la formulation $P - Dsj$ consiste en une structure matricielle en bloc, elle nous aide à décomposer le problème par les principes de Danzig-Wolfe. Cette décomposition a entraîné la construction de l'Algorithme Branch-and-Price pour résoudre le problème. Les tests numériques ont prouvé que la force de l'algorithme dépend beaucoup des bonnes stratégies de branchement. En outre, la branche et le prix sont basés sur l'algorithme de génération de colonne, il nécessite donc des techniques de stabilisation appropriées. Nous avons fourni une politique de branchement efficace pour les variables SOS1 et appliqué la nouvelle borne duales pour stabiliser la procédure de génération de colonne. À la recherche d'une méthode de résolution plus rapide, une famille d'heuristiques constructives est présentée. L'heuristique s'est révélée dédiée au problème ACPV 2. Cela prouve une excellente performance et une réponse rapide au problème de la taille de la vie réelle.

À la fin de cette thèse, nous concevons une méthode de rééchelonnement prédictive-réactive qui peut faire face à l'incertitude et à l'événement en temps réel. Par ce moyen, on peut relier tout le travail théorique à une application réalisable. La prévision basée sur la simulation (planification prédictive) aide à stabiliser le réseau électrique en réduisant le facteur de diversité du stationnement de charge à moins de 20 % selon notre étude de cas. Il a également optimisé le coût d'utilisation de l'énergie en décidant d'un bon abonnement de puissance pour le stationnement et le temps de démarrage quotidien pour la procédure de facturation. La planification prédictive-réactive est concrétisée par le déploiement d'un logiciel de planification planificateur autonome. La structure du logiciel est conçue pour être compatible avec l'état matériel et logiciel existant de l'entreprise et prouver une fonction de charge intelligente au point de charge. Ce logiciel est un avantage concurrentiel pour l'entreprise, alors que nous avons été l'un des premiers en France à fournir un service de recharge intelligent VE.

Avant de terminer cette thèse, nous indiquons notre point de vue, point par point selon chaque contribution.

Tout d'abord, l'étude de la formulation devrait inclure un cas plus général, où les tâches sont autorisées à avoir des interruptions fixes de k. Cette formulation améliorerait la valeur objective tout en maintenant une marge de sécurité acceptable pour la batterie.

Deuxièmement, une heuristique de plongée devrait être conçue pour rendre l'algorithme de dérivation et de prix plus dédié au problème de planification. La branche et le prix sont génériques ; nous pourrions modifier les contraintes locales sans redéfinir un nouvel algorithme. Ainsi, un Branch-and-Price plus rapide devrait ouvrir plus de possibilités pour résoudre de nombreux problèmes dérivés différents de ACPV 2.

Troisièmement, les HLA sont basés sur des éléments consistant en des caractères aléatoires. Nous recherchons actuellement une méthode de recherche adaptative qui peut améliorer l'efficacité de chaque secousse (destruction et construction) de la solution. En outre, la complexité de HLA dépend beaucoup de la complexité de GBF (la phase initiale de la construction de la solution). Par conséquent, une recherche restreinte plus efficace réduirait l'espace de recherche de GBF.

Quatrièmement, nous considérons la construction d'un problème de planification robuste, car les distributions aléatoires des entrées sont connues.

Enfin, une version améliorée du logiciel actuel qui peut considérer des événements plus sophistiqués sont également nos travaux futurs. Les travaux réalisés dans cette thèse sont sujets à plusieurs publications : [51],[56],[52],[54],[55]

Bibliographie

[1] Tobias Achterberg, Thorsten Koch, and Alexander Martin. Branching rules revisited. *Operations Research Letters*, 33(1) :42–54, 2005.

[2] Christopher O Adika and Lingfeng Wang. Smart charging and appliance scheduling approaches to demand side management. *International Journal of Electrical Power & Energy Systems*, 57 :232–240, 2014.

[3] Ravindra K Ahuja, Özlem Ergun, James B Orlin, and Abraham P Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123 (1) :75–102, 2002.

[4] Ali T Al-Awami, Eric Sortomme, Ghous Muhammad Asim Akhtar, and Samy Faddel. A voltage-based controller for an electric-vehicle charger. *IEEE Transactions on Vehicular Technology*, 65(6) :4185–4196, 2016.

[5] Monica Alonso, Hortensia Amaris, Jean Gardy Germain, and Juan Manuel Galan. Optimal charging scheduling of electric vehicles in smart grids by heuristic algorithms. *Energies*, 7(4) :2449–2475, 2014.

[6] Jorge Nájera Álvarez, Katarina Knezović, and Mattia Marinelli. Analysis and comparison of voltage dependent charging strategies for single-phase electric vehicles in an unbalanced danish distribution grid. In *Proceedings of the 51st International Universities Power Engineering Conference*. IEEE, 2016.

[7] Ph Baptiste, Claude Le Pape, and Wim Nuijten. Satisfiability tests and time-bound adjustmentsfor cumulative scheduling problems. *Annals of Operations research*, 92 : 305–333, 1999.

[8] Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. Branch-and-price : Column generation for solving huge integer programs. *Operations research*, 46(3) :316–329, 1998.

[9] Olivier Beaumont and Loris Marchal. A normal form for scheduling work preserving malleable tasks and its applications. *hal. inria, fr*, 2011.

[10] Olivier Beaumont, Nicolas Bonichon, Lionel Eyraud-Dubois, and Loris Marchal. Minimizing weighted mean completion time for malleable tasks scheduling. In *Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, pages 273–284. IEEE, 2012.

[11] Jacek Blazewicz, J. K. Lenstra, and A. H. G. Rinnooy Kan. Scheduling subject to resource constraints : classification and complexity. *Discrete Applied Mathematics*, 5 (1) :11–24, January 1983.

[12] Jacek Blazewicz, Maciej Machowiak, Jan Weglarz, Mikhail Y. Kovalyov, and Denis Trystram. Scheduling Malleable Tasks on Parallel Processors to Minimize the Makespan. *Annals of Operations Research*, 129(1-4) :65–80, July 2004.

[13] Jacek Blazewicz, Mikhail Y Kovalyov, Maciej Machowiak, Denis Trystram, and Jan Weglarz. Preemptable malleable task scheduling problem. *IEEE Transactions on Computers*, 55(4) :486–490, 2006.

[14] Jacek Blazewicz, TC Edwin Cheng, Maciej Machowiak, and Ceyda Oguz. Berth and quay crane allocation : a moldable task scheduling model. *Journal of the Operational Research Society*, 62(7) :1189–1197, 2011.

[15] André Brinkmann, Peter Kling, Friedhelm Meyer auf der Heide, Lars Nagel, Sören Riechers, and Tim Süß. Scheduling Shared Continuous Resources on Many-cores. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '14, pages 128–137, New York, NY, USA, 2014. ACM.

[16] Jacques Carlier and Eric Pinson. Jackson's pseudo-preemptive schedule and cumulative scheduling problems. *Discrete Applied Mathematics*, 145(1) :80–94, 2004.

[17] P Cazzola and M Gorner. Global EV outlook 2016 - beyond one million electric cars. Technical report, International Energy Agency, 2016.

[18] Mark Cieliebak, Thomas Erlebach, Fabian Hennecke, Birgitta Weber, and Peter Widmayer. Scheduling with release times and deadlines on a minimum number of machines. *Exploring New Frontiers of Theoretical Informatics*, pages 209–222, 2004.

[19] SAE Hybrid Committee et al. Sae charging configurations and ratings terminology, 2011.

[20] Jacques Desrosiers and Marco E. Lübbecke. *A Primer in Column Generation*, pages 1–32. Springer US, Boston, MA, 2005.

[21] Pierre-François Dutot, Grégory Mounié, and Denis Trystram. Scheduling parallel tasks : Approximation algorithms. *Handbook of Scheduling : Algorithms, Models, and Performance Analysis*, pages 26–1, 2004.

[22] Direction Commerce EDF. Price list of electricity supply offer. Available at `https://particulier.edf.fr/content/dam/2-Actifs/Documents/Offres/Grille_prix_Tarif_Bleu.pdf` (2016/08/24), 2016.

[23] Samy Faddel, Ali T Al-Awami, and MA Abido. Fuzzy optimization for the operation of electric vehicle parking lots. *Electric Power Systems Research*, 145 :166–174, 2017.

[24] Dominique Feillet. A tutorial on column generation and branch-and-price for vehicle routing problems. *4or*, 8(4) :407–424, 2010.

[25] Thomas A Feo and Mauricio GC Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2) :109–133, 1995.

[26] Gerald Gamrath. *Generic branch-cut-and-price*. PhD thesis, Technische Universitt, Berlin, 2010.

[27] Gerald Gamrath and Marco E Lübbecke. Experiments with a generic dantzig-wolfe decomposition for integer programs. In *Experimental algorithms*, pages 239–252. Springer, 2010.

[28] Mateusz Gorczyca and Adam Janiak. Resource level minimization in the discrete – continuous scheduling. *European Journal of Operational Research*, 203(1) :32–41, 2010.

[29] Mateusz Gorczyca, Adam Janiak, Wladyslaw Janiak, and Marcin Dymański. Makespan optimization in a single-machine scheduling problem with dynamic job ready times—Complexity and algorithms. *Discrete Applied Mathematics*, 182 :162–168, 2015.

[30] Ronald L Graham, Eugene L Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling : a survey. *Annals of discrete mathematics*, 5 :287–326, 1979.

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

[32] John N Hooker. A principled approach to mixed integer/linear problem formulation. *Operations Research and Cyber-Infrastructure*, pages 79–100, 2009.

[33] Wenqi Huang, Duanbing Chen, and Ruchu Xu. A new heuristic algorithm for rectangle packing. *Computers & Operations Research*, 34(11) :3270–3280, 2007.

[34] Shinji Imahori and Mutsunori Yagiura. The best-fit heuristic for the rectangular strip packing problem : An efficient implementation and the worst-case approximation ratio. *Computers & Operations Research*, 37(2) :325–333, 2010.

[35] Emil B Iversen, Juan M Morales, and Henrik Madsen. Optimal charging of an electric vehicle using a markov decision process. *Applied Energy*, 123 :1–12, 2014.

[36] A Janiak. Minimization of the blooming mill standstills-mathematical model, suboptimal algorithms. *Mechanika*, 8(2) :37–49, 1989.

[37] Klaus Jansen and Hu Zhang. Scheduling malleable tasks with precedence constraints. *Journal of Computer and System Sciences*, 78(1) :245–259, 2012.

[38] Piotr Jedrzejowicz and Aleksander Skakovski. Island-based Differential Evolution Algorithm for the Discrete-continuous Scheduling with Continuous Resource Discretisation. *Procedia Computer Science*, 35 :111–117, 2014.

[39] Joanna Józefowska and Jan Weglarz. On a methodology for discrete–continuous scheduling. *European Journal of Operational Research*, 107(2) :338–353, 1998.

[40] Joanna Jozefowska, Marek Mika, Rafal Rozycki, Grzegorz Waligóra, and Jan Weglarz. Local search metaheuristics for discrete – continuous scheduling problems. *European Journal of Operational Research*, 107(2) :354–370, June 1998.

[41] Joanna Jozefowska, Marek Mika, Rafał Rozycki, Grzegorz Waligóra, and Jan Weglarz. A heuristic approach to allocating the continuous resource in discrete – continuous scheduling problems to minimize the makespan. *Journal of Scheduling*, 5(6), November 2002.

[42] Joanna Józefowska, Grzegorz Waligóra, and Jan Weglarz. Tabu list management methods for a discrete–continuous scheduling problem. *European Journal of Operational Research*, 137(2) :288–302, 2002.

[43] Joanna Jozefowska, Grzegorz Waligóra, and Jan Weglarz. A Performance Analysis of Tabu Search for Discrete-Continuous Scheduling Problems. In *Metaheuristics : Computer Decision-Making*, number 86 in Applied Optimization, pages 385–404. Springer US, 2003.

[44] Mohammad M Karbasioun, Gennady Shaikhet, Evangelos Kranakis, and Ioannis Lambadaris. Power strip packing of malleable demands in smart grid. In *Communications (ICC), 2013 IEEE International Conference on*, pages 4261–4265. IEEE, 2013.

[45] Ed Klotz and Alexandra M Newman. Practical guidelines for solving difficult mixed integer linear programs. *Surveys in Operations Research and Management Science*, 18 (1) :18–32, 2013.

[46] Junghoon Lee, Hye-Jin Kim, Gyung-Leen Park, and Hongbeom Jeon. Genetic algorithm-based charging task scheduler for electric vehicles in smart transportation. In *Asian Conference on Intelligent Information and Database Systems*, pages 208–217. Springer, 2012.

[47] Jeff T Linderoth and Martin WP Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, 11(2) :173–187, 1999.

[48] Jonas Maasmann, Christoph Aldejohann, Willi Horenkamp, Michael Kaliwoda, and Christian Rehtanz. Charging optimization due to a fuzzy feedback controlled charging

algorithm. In *Power Engineering Conference (UPEC), 49th International Universities*, pages 1–6. IEEE, 2014.

[49] Rabia Nessah, Chu Chengbin, and Farouk Yalaoui. An exact method for Pm/sds, ri/σni. *Computers & Operations Research*, 34(9) :2840–2848, 2007.

[50] Rabia Nessah, Farouk Yalaoui, and Chengbin Chu. A branch-and-bound algorithm to minimize total weighted completion time on identical parallel machines with job release dates. *Computers & Operations Research*, 35(4) :1176–1190, April 2008.

[51] Nhan-Quy Nguyen, Farouk Yalaoui, Lionel Amodeo, Hicham Chehade, and Pascal Toggenburger. Solving a malleable jobs scheduling problem to minimize total weighted completion times by mixed integer linear programming models. In *Asian Conference on Intelligent Information and Database Systems*, pages 286–295. Springer, 2016.

[52] Nhan-Quy Nguyen, Farouk Yalaoui, Lionel Amodeo, Hicham Chehade, and Pascal Toggenburger. A branch-and-price approach to solving a discrete malleable jobs scheduling problem with time-varying resource constraints. *European Journal of Operational Research, manuscript submitted*, 2017.

[53] Nhan-Quy Nguyen, Farouk Yalaoui, Lionel Amodeo, Hicham Chehade, and Pascal Toggenburger. Reactive rescheduling method for electric vehicles charging in dedicated residential zone parking. In *IEEE Symposium Series on Computational Intelligence, manuscript submitted*. IEEE, 2017.

[54] Nhan-Quy Nguyen, Farouk Yalaoui, Lionel Amodeo, Hicham Chehade, and Pascal Toggenburger. Predictive baseline schedule for electrical vehicles charging in dedicated residential zone parking. *Metaheuristics : Proceeding of the MIC and MAEB 2017 Conferences*, 2017.

[55] Nhan-Quy Nguyen, Farouk Yalaoui, Lionel Amodeo, Hicham Chehade, and Pascal Toggenburger. Electrical vehicle charging coordination algorithms framework, manuscript submitted. In *Collective and computational intelligence applications in energy industry*. Springer, 2017.

[56] Nhan-Quy Nguyen, Farouk Yalaoui, Lionel Amodeo, Hicham Chehade, and Pascal Toggenburger. Total completion time minimization for machine scheduling problem under

time windows constraints with jobs' linear processing rate function. *Computers & Operations Research*, 90 :110–124, 2018.

[57] Fabrice Talla Nobibon, Roel Leus, Kameng Nip, and Zhenbo Wang. Resource loading with time windows. *European Journal of Operational Research*, 244(2) :404–416, 2015.

[58] Eugeniusz Nowicki and Stanisław Zdrzałka. A survey of results for sequencing problems with controllable processing times. *Discrete Applied Mathematics*, 26(2-3) :271–287, 1990.

[59] Michael Angelo A Pedrasa, Ted D Spooner, and Iain F MacGill. Coordinated scheduling of residential distributed energy resources to optimize smart home energy services. *IEEE Transactions on Smart Grid*, 1(2) :134–143, 2010.

[60] Michael Pinedo. In *Scheduling*, chapter 9. Springer, 2015.

[61] David Pisinger and Stefan Ropke. Large neighborhood search. In *Handbook of metaheuristics*, pages 399–419. Springer, 2010.

[62] R Różycki and J Węglarz. Power-aware scheduling of preemptable jobs on identical parallel processors to meet deadlines. *European Journal of Operational Research*, 218 (1) :68–75, 2012.

[63] Rafał Rozycki and Jan Weglarz. Solving a power-aware scheduling problem by grouping jobs with the same processing characteristic. *Discrete Applied Mathematics*, 182 :150–161, February 2015.

[64] Ruslan Sadykov. A dominant class of schedules for malleable jobs in the problem to minimize the total weighted completion time. *Computers & Operations Research*, 39 (6) :1265–1270, 2012.

[65] Dvir Shabtay and Moshe Kaspi. Minimizing the total weighted flow time in a single machine with controllable processing times. *Computers & Operations Research*, 31(13) : 2279–2289, 2004.

[66] Dvir Shabtay and George Steiner. A survey of scheduling with controllable processing times. *Discrete Applied Mathematics*, 155(13) :1643–1666, August 2007.

[67] Dvir Shabtay and George Steiner. Optimal due date assignment and resource allocation to minimize the weighted number of tardy jobs on a single machine. *Manufacturing & Service Operations Management*, 9(3) :332–350, 2007.

[68] James Richard Tebboth. A computational study of dantzig-wolfe decomposition. *University of Buckingham*, 2001.

[69] U.S. Department of Energy (DOE) and North Carolina State Energy Office. Real-World Charging Behavior at the Workplace. Plug-in Electric Vehicle Consumer Usage Study, Advanced Energy leveraged a U.S. Department of Energy (DOE), 2014.

[70] Stijn Van de Vonder, Erik Demeulemeester, and Willy Herroelen. A classification of predictive-reactive project scheduling procedures. *Journal of Scheduling*, 10(3) :195–207, 2007.

[71] François Vanderbeck. On dantzig-wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, 48 (1) :111–128, 2000.

[72] François Vanderbeck. Branching in branch-and-price : a generic scheme. *Mathematical Programming*, 130(2) :249–294, 2011.

[73] François Vanderbeck and Martin WP Savelsbergh. A generic view of dantzig–wolfe decomposition in mixed integer programming. *Operations Research Letters*, 34(3) : 296–306, 2006.

[74] François Vanderbeck and Laurence A Wolsey. An exact algorithm for IP column generation. *Operations Research Letters*, 19(4) :151–159, 1996.

[75] RG Vickson. Two single machine sequencing problems involving controllable job processing times. *AIIE transactions*, 12(3) :258–262, 1980.

[76] Guilherme E Vieira, Jeffrey W Herrmann, and Edward Lin. Rescheduling manufacturing systems : a framework of strategies, policies, and methods. *Journal of Scheduling*, 6(1) :39–62, 2003.

[77] Juan Pablo Vielma. Mixed integer linear programming formulation techniques. *SIAM Review*, 57(1) :3–57, 2015.

[78] Grzegorz Waligóra. Tabu search for discrete–continuous scheduling problems with heuristic continuous resource allocation. *European Journal of Operational Research*, 193 (3) :849–856, 2009.

[79] Grzegorz Waligóra. Heuristic approaches to discrete-continuous project scheduling problems to minimize the makespan. *Computational Optimization and Applications*, 48 (2) :399–421, August 2010.

[80] Jan Weglarz. Time-Optimal Control of Resource Allocation in a Complex of Operations Framework. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-6(11) :783–788, November 1976.

[81] Laurence A Wolsey. *Integer programming*. Wiley, 1998.

[82] Kailiang Xu, Zuren Feng, and Keliang Jun. A tabu-search algorithm for scheduling jobs with controllable processing times on a single machine to meet due-dates. *Computers & Operations Research*, 37(11) :1924–1938, 2010.

[83] Farouk Yalaoui and Chengbin Chu. Parallel machine scheduling to minimize total tardiness. *International Journal of Production Economics*, 76(3) :265–279, 2002.

[84] Murat Yilmaz and Philip T Krein. Review of battery charger topologies, charging power levels, and infrastructure for plug-in electric and hybrid vehicles. *IEEE Transactions on Power Electronics*, 28(5) :2151–2169, 2013.

Nhan Quy NGUYEN

Doctorat : Optimisation et Sûreté des Systèmes

Année 2017

Optimisation de l'ordonnancement de recharges des véhicules électriques

Notre travail de recherche traite de la problématique de l'ordonnancement de recharge des véhicules électriques (VE). La variation de la puissance totale disponible pour charger des véhicules, les contraintes de comportement des utilisateurs et l'incertitude des demandes énergétiques journalières demandent un ordonnancement efficace et sécurisé.

Nous avons défini cinq configurations industrielles : ACPF (1,2) et ACPV (1a, 1b et 2) qui correspondent chacune à un ensemble de contraintes techniques. Les études sur les formulations, dont une conjonctive et une disjonctive, reposent sur l'analyse de la force de leurs relaxation-LP. La forme matricielle de la formule mathématique est composée d'une matrice partitionnée, qui est décomposable par le principe de Dantzig-Wolfe. Cette dernière nous permets de développer un algorithme de type Branch-and-Price pour la résolution exacte du problème. Une heuristique constructive déterministe a ensuite été conçue pour l'allocation de la ressource, qui se trouve très efficace : une résolution rapide (moins d'une seconde) pour un parking d'une trentaine VEs. Finalement, pour implémenter tous les algorithmes dans le microprocesseur, et pour établir un modèle prévisionnel et un ordonnancement en temps réel, nous avons créé un planificateur autonome, qui se base sur le réordonnancement prédictif-réactif.
Les recherches effectuées font partie des problèmes de raisonnement énergétique. Elles possèdent donc la capacité de se combiner avec d'autres travaux, notamment le problème de smart grid.

Mots clés : optimisation combinatoire - heuristique - véhicules électriques - ordonnancement (gestion) – décomposition (méthode mathématique).

Electric Vehicle Charging Scheduling Optimisation

Our research deals with the problem of the charging scheduling of electric vehicles (EV). The variation in the total power available to load vehicles, user the behaviour constraints and the uncertainties of daily energy demands require an efficient and secure scheduling.

We defined five industrial configurations: ACPF (1,2) and ACPV (1a, 1b and 2), each of which corresponds to a set of technical constraints. Studies on formulations, including a conjunctive and a disjunctive, are based on the analysis of the strength of their LP-relaxation. The matrix form of the mathematical formula is composed of a partitioned matrix, which is decomposable by the Dantzig-Wolfe principles. The latter allows us to develop a Branch-and-Price Algorithm for the exact solution of the problem. A deterministic constructive heuristic was then designed for the allocation of the resource, which is very efficient: a quick resolution (less than a second) for a car park with about thirty EVs. Finally, to implement all algorithms in the microprocessor, and to establish a forecasting model and an online scheduling, we have created a stand-alone scheduler, based on the predictive-reactive rescheduling.

The research carried out is part of the problems of energy reasoning. They, therefore, can combine with other works, including the smart grid problems.

Keywords: combinatorial optimization - heuristic - electric vehicles -production scheduling - decomposition method.

Thèse réalisée en partenariat entre :

anrt association nationale recherche technologie — Cifre Conventions — park'n plug — utt UNIVERSITÉ DE TECHNOLOGIE TROYES

Ecole Doctorale "Sciences et Technologies"