École doctorale Informatique, Télécommunications et électronique (Paris)

Sorbonne Université

Laboratoire d'Informatique de Paris 6 (Équipe Recherche Opérationnelle)

Département OSIRIS - EDF R&D

Thèse de doctorat en informatique

Aspects combinatoires du Unit Commitment Problem

Présentée et soutenue publiquement le 14 novembre 2018 par

Cécile Rottner

Jury:

Christian Artigues Directeur de recherche au LAAS–CNRS, Toulouse, France	Rapporteur
Volker Kaibel Professeur à Otto-von-Guericke-Universität Magdeburg, Allemagne	Rapporteur
Philippe Chrétienne Professeur à Sorbonne Université, Paris, France	Examinateur
Ivana Ljubic Professeur à l'ESSEC Business School, Cergy, France	Examinatrice
Pascale Bendotti Ingénieur-Chercheur à EDF R&D, Palaiseau, France	Co-encadrante
Pierre Fouilhoux Maître de Conférences à Sorbonne Université, Paris, France	Directeur de thèse

ACKNOWLEDGEMENTS

First I would like to thank Christian Artigues and Volker Kaibel for making me the honor of reviewing my thesis. I also thank Philippe Chrétienne and Ivana Ljubic, for generously accepting to be part of my thesis's jury.

Then I would like to express my deepest gratitude to my advisors Pascale and Pierre, for all their stimulating guidance, support, availability and knowledge, and also for their continuous help to make me succeed not only in my thesis but also in my professional life. They share a great responsibility for making these three years an amazing time. I would like to thank also Christoph Dürr, in particular for nice and fruitful discussions on complexity matters.

Of course these three years would not have been so great without my colleagues from EDF R&D and LIP6, with whom I also learned a lot, in particular many board games. We enjoyed a lot of things together: lunches, coffees, riddles and puzzles, cakes, movie and book talks, beers, coinches, Moroccan, Italian and French trips, office gardening, fighting for paper recycling, and sometimes even work. I wish to thank in particular those who patiently provided me with useful advice and knowledge, and who gave me the chance to pursue the adventure at EDF R&D.

I am also grateful to all the inspiring teachers I had at the MPRO, ENSTA, UT Austin, and (a long time ago) in "prépa" and high school.

Finally, I would like to thank my friends and family, including those who came to support me at the defense, and in particular: the CNRTL security council (and counterparts from Brittany and Grenoble); the Free Catalonia working group; the $\mathbb{K}$-sos collective; the Zimmi's; my parents; my sister Diane, who is like a sister to me; and Théo, for his unfailing and high-dimensional support.

EXTENDED ABSTRACT

The Unit Commitment Problem (UCP) is a central power management problem at EDF. The core problem of the UCP, called the Min-up/min-down Unit Commitment Problem (MUCP), is to find a minimum-cost production plan on a discrete time horizon for a set of units producing electric power. At each time period, the total production has to meet a forecast demand. Each unit must satisfy minimum up-time and down-time constraints besides featuring production and start-up costs. We analyze how the MUCP complexity evolves with respect to the number n of units and T of time periods. A classical reduction from the knapsack problem shows that the MUCP is NP-hard in the ordinary sense even for $T = 1$. When either a unitary cost or amount of power is considered, the MUCP is polynomial for $T = 1$ and is shown to be strongly NP-hard for arbitrary T.

Some polyhedral aspects of the MUCP are investigated and extend literature results that were limited to one production unit. We define up-set inequalities as the MUCP equivalent of extended cover inequalities from the 0-1 knapsack polytope. We introduce interval up-set inequalities, a new class of valid inequalities, generalizing both up-set inequalities and min-up constraints. Characterization of validity and facet defining cases are given. An efficient Branch & Cut algorithm is devised.

Symmetries arising in the solution set of a given integer linear program can impair the solution process. We define sub-symmetries, as symmetries arising from a solution subset. We focus on integer linear programs whose solutions are binary matrices and whose (sub-)symmetry groups are symmetric groups acting on (sub-)columns. We propose a general framework to handle sub-symmetries in such problems, first by showing how to select one representative for each class of symmetrical solutions, given that several sub-symmetry groups are simultaneously considered. Second, we propose two symmetry-breaking techniques removing all non-representative solutions. The first technique is an orbitopal fixing algorithm for the full orbitope, defined as the convex hull of binary matrices with lexicographically nonincreasing columns. The idea is to determine all the variables whose values are fixed in the intersection of an hypercube face with the full orbitope. We introduce a dynamic variant of this orbitopal fixing algorithm, where the lexicographical order follows the branching decisions occurring along the B&B search. The second proposed technique is based on sub-symmetry breaking inequalities, by introducing one additional variable per sub-symmetry group considered. In the MUCP case, no additional variable is needed to derive such

symmetry-breaking inequalities, which can also be further lifted. Experimental results on MUCP instances show that the proposed techniques outperform state-of-the-art symmetry-breaking techniques.

Finally, we compare the dual bounds obtained with various Dantzig-Wolfe decomposition structures for the MUCP. In particular, we show that the dual bound obtained by dualization of the time-coupling constraints is better than the bound provided by Cplex's own cuts. This bound is further improved by interval up-set inequalities. The resulting Branch & Price & Cut features promising exact and heuristic performances.

TABLE OF CONTENTS

Page

Introduction **9**

1 Definitions and state-of-the-art **13**

1.1 Combinatorial optimization 13

1.1.1 Polyhedral combinatorics 13

1.1.2 Branch & Bound, Branch & Cut 15

1.1.3 Lagrangian and Dantzig-Wolfe decompositions 16

1.2 The Unit Commitment Problem 17

1.2.1 Operational constraints and costs of the UCP 17

1.2.2 Resolution of the UCP by Lagrangian relaxation 19

1.2.3 The Min-up/min-down Unit Commitment Problem 20

1.2.4 Instances 22

1.2.5 ILP formulations of the Min-up/min-down Unit Commitment Problem . . 23

1.2.6 Polyhedral studies of the 1-unit UCP 30

1.2.7 ILP formulations of the UCP 31

2 On the complexity of the Unit Commitment Problem **33**

2.1 The UCP is strongly NP-hard 34

2.2 The 1-period MUCP is weakly NP-hard 35

2.3 The IMUCP is polynomial when n is fixed 36

2.4 NP-hard special cases of the MUCP 38

2.4.1 The unit-cost MUCP is NP-hard 38

2.4.2 The unit-power MUCP is strongly NP-hard 39

2.5 The P-IMUCP is strongly NP-hard 42

I Polyhedral combinatorics of the MUCP **47**

3 The Min-up/min-down Unit Commitment polytope **49**

3.1 Comparison of demand-coupling formulations 50

3.1.1 Comparison of demand-coupling formulations for the n-unit MUCP 50
3.1.2 Integrality of (F^1-*Flow*) 52
3.2 Polyhedral study 55
3.3 Rank of unit subsets 59
3.4 Valid inequalities 62
3.4.1 Up-set inequalities 62
3.4.2 Interval Up-Set inequalities 66
3.4.3 Generalized interval up-set inequalities 70
3.5 Facial study for interval up-set inequalities 72
3.5.1 Necessary facet conditions in $P^n_{x,u}$ 72
3.5.2 Facet characterization in $P^n_{x,u}(\mathscr{I})$ 73

4 Branch & Cut for the MUCP **77**
4.1 Separation 77
4.1.1 Separation of up-set inequalities 77
4.1.2 Separation of interval up-set inequalities 79
4.2 Experimentation 79

II Breaking symmetries and sub-symmetries **89**

5 Symmetries and sub-symmetries **91**
5.1 Definitions 91
5.2 State-of-the-art of generic symmetry-breaking techniques 93
5.2.1 Symmetry-breaking inequalities 94
5.2.2 Symmetry-breaking polytopes 95
5.2.3 Pruning by isomorphism in Branch & Bound 95
5.2.4 Orbital branching 96
5.2.5 Variable fixing in symmetry-breaking polytopes 97
5.2.6 Variable aggregation 98
5.3 State-of-the-art for the symmetric group case 98
5.3.1 Symmetry-breaking inequalities 99
5.3.2 Modified orbital branching 99
5.3.3 Orbitopes and orbitopal fixing 100
5.4 State-of-the-art of symmetry-breaking for the UCP 101
5.5 Sub-symmetries and sub-orbitopes 104
5.5.1 Sub-symmetries 104
5.5.2 Full sub-orbitopes 108
5.5.3 Sub-symmetries in the MUCP 109

6 Orbitopal fixing for the full (sub-)orbitope **111**
6.1 Intersection with the full orbitope . . . 112
6.1.1 Two matrix sequences . . . 113
6.1.2 Determining $I_0^\star$ and $I_1^\star$ . . . 116
6.2 Static and dynamic orbitopal fixing . . . 117
6.2.1 Static orbitopal fixing . . . 117
6.2.2 Dynamic orbitopal fixing . . . 118
6.2.3 Orbitopal fixing in the full sub-orbitope . . . 120
6.3 Orbitopal fixing for the MUCP . . . 120
6.4 Experimental results for the MUCP . . . 121
6.4.1 Instances . . . 122
6.4.2 Static and dynamic orbitopal fixing . . . 123
6.4.3 Comparison of Cplex, MOB, DOF and DOF-S . . . 127

7 Sub-symmetry breaking inequalities **131**
7.1 Definition and validity . . . 132
7.2 Full symmetry-breaking sufficient condition . . . 133
7.3 Application to the symmetric group case . . . 135
7.4 Application to the Unit Commitment Problem . . . 136
7.4.1 Sub-symmetry-breaking inequalities for the MUCP . . . 137
7.4.2 Sub-symmetry-breaking inequalities for the ramp-constrained MUCP . . . 138
7.5 Experimental results . . . 139
7.5.1 Experimental settings . . . 139
7.5.2 Instances . . . 140
7.5.3 Results for the MUCP . . . 140
7.5.4 Results for the ramp-constrained MUCP . . . 141

III Branch & Price for the MUCP **149**

8 Decomposition structure **151**
8.1 Motivations . . . 151
8.2 Unit subset decomposition of the IMUCP . . . 152
8.2.1 Dualization of production constraints . . . 153
8.2.2 Granularity of the unit-subset decomposition . . . 155
8.2.3 Start-up decomposition . . . 156
8.2.4 Resolution of the subproblems . . . 157
8.3 Time decomposition of the IMUCP . . . 158
8.3.1 Dualization of production constraints . . . 159
8.3.2 Time decomposition with interval up-set inequalities . . . 159

8.3.3 Resolution of the subproblem . . . 159
8.4 Experimental results relative to dual bounds . . . 160
8.4.1 Experimental settings . . . 160
8.4.2 Impact of intra-site constraints . . . 162
8.4.3 Granularity of the unit subset decomposition . . . 163
8.4.4 Start-up decomposition . . . 165
8.4.5 Time decomposition . . . 166
8.4.6 Time decomposition with interval up-set inequalities . . . 167
8.5 Experimental results relative to Branch & Price & Cut . . . 169
8.5.1 First results on small-size instances . . . 170
8.5.2 Results on larger instances . . . 170
8.6 Experimental results relative to Price & Branch heuristic . . . 171
8.7 Perspectives on symmetry-breaking in Dantzig-Wolfe reformulations . . . 172
8.7.1 Handling symmetries in unit-subset decompositions . . . 172
8.7.2 Handling symmetries in time decomposition . . . 174
8.8 Conclusion . . . 175
8.9 Experimental tables . . . 176

Conclusions **203**
Experimental summary . . . 203
Conclusion and perspectives . . . 207

Bibliography **211**

INTRODUCTION

The Unit Commitment Problem (UCP) commonly identifies the real-world daily power generation problem for electric companies. EDF (Électricité de France), one of the world's largest producer of electricity, manages a mix of heterogeneous production units (nuclear power plants, hydropower plants, fuel oil and gas turbines, gas combined cycle power plants, ...). The problem is then to fulfill an hourly power demand on a two-day time horizon, deciding when each production unit is up and which quantity of power it produces. Each unit possesses its own hard-to-manage technical constraints, for example, when the unit is up, its production must be above some minimal production limit.

Various aspects of the UCP have been extensively studied in the literature. Historically, the problem has been solved using methods arising from continuous optimization, such as Lagrangian relaxation of the demand constraint. Such a technique often leads to solutions far from feasibility. Therefore an alternative is to use integer linear programming (ILP) to solve the problem. The efficiency of these techniques highly relies on the problem's structure, thus the combinatorial aspects of the problem must be taken into account when handling difficult UCP instances.

In this thesis, we study some combinatorial aspects of the UCP by focusing on the Min-up/Min-down Unit Commitment Problem (MUCP), which is the core structure of the thermal UCP, *i.e.*, the UCP featuring nuclear, fuel oil, coal and gas units. Besides the production limits, an MUCP unit has to satisfy the min-up/min-down constraints, *i.e.*, it must remain up (resp. down) long enough after start-up (resp. shut-down). In Chapter 1, we precise the definition and review the combinatorial aspects of the MUCP together with some additional technical constraints.

The UCP is known to be NP-hard even for a single-time-period horizon, by a direct reduction from the 0-1 knapsack problem. Proving that the arbitrary-size horizon UCP is strongly NP-hard, the complexity analysis in Chapter 2 highlights that the combinatorial difficulty of the MUCP lies also in the dynamic coupling of power demand constraints.

As the polyhedral analyses conducted in the literature only concern the single production unit UCP, they do not deal with this difficulty arising from the time coupling of multiple production units.

In Part I (Chapters 3 and 4), we study the polytope of the MUCP involving multiple units. We show that several classical MUCP formulations have the same relaxation value, thus our polyhedral analysis is based on the formulation featuring variables corresponding to natural decisions for the MUCP. We derive new valid inequalities, namely interval up-set inequalities, precisely capturing both dynamic and knapsack features of the MUCP. We characterize the cases in which these inequalities define facets of the associated polytope, in order to get theoretical conditions specifying when an interval up-set inequality is the strongest possible. The derived Branch & Cut algorithm appears to improve the classical MUCP formulation.

Symmetries arising from combinatorial aspects of the UCP dramatically impair its resolution by ILP techniques. Symmetry-related issues in ILP featuring all-(sub-)column permutation symmetries are addressed in Part II (Chapters 5, 6 and 7).

We review existing symmetry-breaking techniques in Chapter 5. We refer to symmetries arising in solution subsets of a given integer program as *sub-symmetries*. Sub-symmetries may not arise in the full solution set. However, this observation is not exploited in practice by existing symmetry-breaking techniques, as this would imply to compute the problem's sub-symmetries at each node of the Branch & Bound tree, which is computationally prohibitive. In many applications, sub-symmetries can be easily obtained from the problem's structure, and therefore do not need to be computed at each node. We propose in Chapter 5 a theoretical framework to handle such sub-symmetries. In particular, we consider how to select one representative of each class of symmetrical solutions, when multiple symmetry groups are considered.

If some of the existing symmetry-breaking techniques rely on the addition of inequalities removing symmetrical solutions from the feasible set, others are based on pruning actions in the Branch & Bound tree. In the latter case, the size of the linear program solved at each node does not increase.

We consider integer linear programs whose solutions are binary matrices and whose symmetry groups are symmetric groups acting on columns. Existing symmetry-breaking techniques for such problems remove all symmetrical solutions from the feasible set only at the expense of not being flexible, *i.e.*, imposing restrictions on the branching disjunctions. Orbitopal fixing, as introduced in the literature, is a flexible pruning technique designed to break all all-column-permutation symmetries in the special case of partitioning (resp. packing) problems whose solution matrices feature exactly (resp. at most) one 1-entry in each row.

Such all-column permutation symmetries arise in the UCP. However, no particular restriction on the number of 1-entries in each solution row applies in this case. In Chapter 6, we propose an orbitopal fixing algorithm to break all-column-permutation symmetries in any integer program whose solutions feature an arbitrary number of 1-entries in each row. This technique is flexible and also removes all symmetrical solutions from the feasible set. We show that this orbitopal fixing algorithm can be used to break both symmetries and sub-symmetries, in all integer linear

programs whose (sub-)symmetry groups are symmetric groups acting on (sub-)columns of the solution matrix.

Symmetries and sub-symmetries in such programs can also be tackled by the sub-symmetry-breaking inequalities we introduce in Chapter 7. These inequalities feature at most one additional variable per sub-symmetry group considered. In the special case of the MUCP, even stronger alternate inequalities can be derived with no additional variable.

The two proposed symmetry-breaking techniques apply not only to the MUCP but also to all problems whose (sub-)symmetry groups are symmetric groups acting on (sub-)columns. Experimental results carried out on MUCP instances show the efficiency of each proposed symmetry-breaking technique, namely orbitopal fixing for the full sub-orbitope and sub-symmetry-breaking inequalities. Interestingly, on ramp-constrained MUCP instances, sub-symmetry-breaking inequalities outperform all state-of-the-art symmetry-breaking formulations.

The UCP features several structures which can be exploited in a decomposition framework. The question is then on which structure should the decomposition be based, and which techniques should be used, in order to best handle the combinatorial aspects of the problem. This would suggest an alternative approach to the classical Lagrangian relaxation of the demand constraint. In Part III (Chapter 8), we analyze various decomposition structures for the IMUCP, an MUCP variant featuring more coupling constraints. Valid inequalities and (sub-)symmetry-breaking techniques are useful to handle some combinatorial aspects of the UCP. These techniques can be applied in many contexts, from Branch & Bound to Branch & Price frameworks. We study how the combinatorial techniques presented in Parts I and II, in particular interval up-set inequalities, integrate into these decomposition contexts in order to obtain more efficient tools to solve the real-world UCP.

As a conclusion we provide some perspectives on the theoretical contributions alongside with their experimental impact.

CHAPTER 1

DEFINITIONS AND STATE-OF-THE-ART

1.1 Combinatorial optimization

In this section, we introduce only a few definitions and properties related to combinatorial optimization. For more details, one can refer to [69].

An *optimization problem* consists of finding an optimal solution from a given set of solutions.

In order to exactly solve optimization problems arising in operational research contexts, a large literature of theoretical frameworks and practical methods have been developed, such as combinatorial algorithms based on dominances or on dynamic programming schemes.

An optimization problem can also be expressed as an *integer linear program* (ILP) as follows:

$$\begin{aligned} (P) \quad v = \min_{x \in \mathbb{R}^n} \quad & \boldsymbol{c}x \\ \text{s. t.} \quad & \boldsymbol{A}x \leq \boldsymbol{b} \\ & x_i \in \mathbb{Z} \quad \forall i \in I \end{aligned}$$

where $I \subseteq \{1,...,n\}$, $\boldsymbol{A} \in \mathbb{Q}^{m \times n}$, $\boldsymbol{b} \in \mathbb{Q}^m$ and $\boldsymbol{c} \in \mathbb{R}^n$, n, $m \in \mathbb{N}$.

Such a problem is NP-hard [28] in general. When index set $I = \varnothing$, then problem (P) is a *linear program*, as the solution set is described by linear inequalities only. Linear programs can be solved in polynomial time [46], for example by interior points methods [44]. There also exist other efficient resolution techniques, such as the simplex algorithm [15].

1.1.1 Polyhedral combinatorics

A *polyhedron* $\mathscr{P} \subseteq \mathbb{R}^n$ is the solution set of a finite system of linear inequalities, *i.e.*,

$$\mathscr{P} = \{x \in \mathbb{R}^n \mid \boldsymbol{M}x \leq \boldsymbol{\alpha}\}$$

where $M \in \mathbb{Q}^{m \times n}$, $\alpha \in \mathbb{Q}^m$, and $n, m \in \mathbb{N}$. A *polytope* is a bounded polyhedron.

A solution $x \in \mathscr{P}$ is an *extreme point* of $\mathscr{P}$ if there exist no solutions $x_1, x_2 \in \mathscr{P}$, $x_1 \neq x_2$ such that $x = \frac{1}{2}x_1 + \frac{1}{2}x_2$. The dimension $dim(\mathscr{P})$ of $\mathscr{P}$ is $d-1$, where d is the maximum number of affinely independent points in $\mathscr{P}$. If $dim(\mathscr{P}) = n$ then $\mathscr{P}$ is full-dimensional.

An inequality $ax \leq \beta$ is valid for $\mathscr{P}$ if it is satisfied by all points of $\mathscr{P}$. The *face* F of $\mathscr{P}$ defined by a valid inequality $ax \leq \beta$ is $F = \{x \in \mathscr{P} \mid ax = \beta\}$. If $dim(F) = dim(\mathscr{P}) - 1$ then F is a *facet* of $\mathscr{P}$. An inequality $ax \leq \beta$ is said to be *redundant* with respect to a linear system $Mx \leq \alpha$ if it can be obtained from a linear combination of inequalities $Mx \leq \alpha$.

If $\mathscr{P}$ is a polytope, a minimal description of $\mathscr{P}$ is given by a set of inequalities which are in one to one correspondence with facets of $\mathscr{P}$.

The solution set of any linear program is a polyhedron $\mathscr{P}$. Following from the simplex algorithm, the set of extreme points of $\mathscr{P}$ always contains at least one optimal solution. Therefore, (P) can be reformulated as (LP) $\min\{cx \mid x \in conv(S)\}$, where $conv(S)$ is the convex hull of the *solution set* $S = \{x \mid Ax \leq b,\ x_i \in \mathbb{Z},\ \forall i \in I\}$ of (P).

As $conv(S)$ is a polyhedron [63], it follows that (LP) is a linear program.

The idea of polyhedral combinatorics is to study polyhedron $conv(S)$, in order to get insights into the combinatorial structure of problem (P) and derive efficient algorithms to solve it.

For a given set $S = \{x \in \mathbb{Z}^n \mid Ax \leq b\}$, if $conv(S) = \{x \in \mathbb{R}^n \mid Ax \leq b\} = \mathscr{P}_S$ then polyhedron $\mathscr{P}_S$ is said to be *integral*.

Theorem 1.1 ([19])**.** *A polyhedron P is integral if and only if for any $c \in \mathbb{Z}^n$, if z is finite, then z is an integer, where $z = \max\{cx \mid Ax \leq b\}$.*

A matrix A is *totally unimodular* if for any square submatrix A_C of A, the determinant of A_C is in $\{-1, 0, 1\}$.

Theorem 1.2 ([37])**.** *If A is totally unimodular, then the polyhedron $\{x \mid Ax \leq b\}$ is integral for any integer vector b.*

Consider system of inequalities $Ax \leq b$, where $A \in \mathbb{Z}^{(m,n)}$, $b \in \mathbb{Z}^m$. System $Ax \leq b$ has the *integer decomposition property* if, for any integer k, and $\overline{x} \in \mathbb{Z}^n$ such that $A\overline{x} \leq kb$, there exist x_1, ..., $x_k \in \mathbb{Z}^n$ such that $Ax_{k'} \leq b$, $k' \in \{1, ..., k\}$, and $\overline{x} = x_1 + ... + x_k$.

Theorem 1.3 ([7])**.**

A is totally unimodular $\iff \forall b \in \mathbb{Z}^m$, $Ax \leq b$ has the integer decomposition property.

In general, if a complete description of $conv(S)$ by a system of linear inequalities $Mx \leq b$ is found, the solving time of the associated linear program will depend on the separation algorithm for inequalities $Mx \leq b$. The *separation problem* for $Mx \leq b$ is to find, for any point $\overline{x} \in \mathbb{R}^n$, an inequality in the system $Mx \leq b$ which is not satisfied by $\overline{x}$. If no such inequality is found, then $\overline{x}$ satisfies the whole system $Mx \leq b$. An algorithm solving the separation problem is called

separation algorithm. The cutting plane based method is to iterate a separation algorithm until a solution x satisfying $Mx \leq b$ is found.

Grötschel, Lovász and Schrijver [34] have shown that the (arbitrary large) linear program $Mx \leq b$ can be solved in polynomial time (in the input size of the original problem (P)) by a cutting plane based method if and only if the associated separation algorithm for $Mx \leq b$ is polynomial.

When the underlying optimization problem (P) is NP-hard, a complete description of $conv(S)$ by a system of linear inequalities is unlikely to be reached [45]. However, a partial description of $conv(S)$ can still be useful, in the context of a Branch & Bound algorithm, to obtain good lower bounds on the optimal solution value of (P) and of its subproblems.

1.1.2 Branch & Bound, Branch & Cut

A *Branch & Bound* (B&B) algorithm is to enumerate candidate solutions to (P) by means of a rooted tree, the root corresponding to the full solution set S. The principle of the algorithm is to split recursively the search space in smaller spaces. The algorithm explores branches of this tree, each node representing a subset of the solution set. At each node, a lower and an upper bound on the corresponding subproblem solution value is computed, and if no better solution than the one found by the algorithm so far can be produced, the node is discarded.

The linear relaxation of (P) is the linear program (LR) $v_{LR} = \min\{x \in \mathbb{R}^n \mid Ax \leq b\}$, where the integrality constraints of (P) have been relaxed. The optimal value v_{LR} of (LR) is a lower bound on the optimal value of (P). Classically, at each node of the B&B tree, the lower bound on the solution value is computed by linear relaxation of the corresponding subproblem.

Note that value v_{LR} depends on the inequalities $Ax \leq b$ used to describe solution set S. In order to get a lower bound v_{LR} as close as possible to the optimal integer solution value, one must not only use inequalities $Ax \leq b$, but also valid inequalities $A'x \leq b'$, so that polyhedorn $\{x \in \mathbb{R}^n \mid Ax \leq b,\ A'x \leq b'\}$ is as "close" as possible to $conv(S)$. If such valid inequalities are in exponential number, they cannot be handled in a linear program all at a time. The *Branch & Cut* algorithm is a Branch & Bound where, at each node, once the linear relaxation (LR) is solved, a cutting plane based method finds inequalities in system $A'x \leq b'$ which are not satisfied by solution $\overline{x}$ of (LR). Such inequalities are added to (LR).

For efficiency purpose, valid inequalities $A'x \leq b'$ must be non-redundant. As inequalities defining facets of $conv(S)$ are never redundant, the Branch & Cut algorithm will be more efficient when inequalities $A'x \leq b'$ define facets of $conv(S)$. Moreover the polytopes associated to the B&B nodes are more likely to be integral.

1.1.3 Lagrangian and Dantzig-Wolfe decompositions

Consider an ILP (P') $v = \min_{x\in\mathbb{R}^n} \{ \boldsymbol{c}x \mid \boldsymbol{A}x \geq \boldsymbol{d},\ x \in X \}$, where $\boldsymbol{A} \in \mathbb{Q}^{m\times n}$, $\boldsymbol{d} \in \mathbb{Q}^m$, $\boldsymbol{c} \in \mathbb{R}^n$ and $X \subseteq \mathbb{N}^{n_1} \times \mathbb{R}^{n_2}$, $n = n_1 + n_2$, is a mixed integer set.

The associated *Lagrangian function* θ is defined as follows.

$$\forall \mu \in \mathbb{R}^m_+, \quad \theta(\mu) = \min_{x\in X} L(x,\mu) \quad \text{where} \quad L(x,\mu) = \boldsymbol{c}x + \mu(d - \boldsymbol{A}x)$$

Note that for each $\mu \in \mathbb{R}^m_+$, $\theta(\mu)$ is a lower bound of v. The bound v_D obtained by maximizing θ over $\mu \in \mathbb{R}^m_+$ is called *Lagrangian relaxation* or *dual bound*:

$$v_D = \max_{\mu\in\mathbb{R}^m_+} \theta(\mu).$$

The corresponding LP is called *Lagrangian dual*. As θ is a concave function, there exist efficient algorithms for its maximization, such as subgradient methods [91].

Another approach to maximize θ is to consider $\{x_\pi,\ \pi \in \{1,...,\Pi\}\}$, the set of extreme points of $conv(X)$, and rewrite v_D as

$$v_D = \max_{\mu\in\mathbb{R}^m_+} \min_{\pi\in\{1,...,\Pi\}} (c - \mu\boldsymbol{A})x_\pi + \mu d = \max_{\sigma\in\mathbb{R},\ \mu\in\mathbb{R}^m_+} \left\{\sigma + \mu d \mid \sigma \leq (c - \mu\boldsymbol{A})x_\pi,\ \pi \in \{1,...,\Pi\}\right\} \tag{1.1}$$

Taking the linear programming dual of the latter problem, the LP obtained is called the *Dantzig-Wolfe master problem*:

$$\begin{aligned} (MP) \quad v_D \quad = \quad \min_{\lambda_\pi\in\mathbb{R}^m_+} \quad & \sum_{\pi=1}^{\Pi} (\boldsymbol{c}x_\pi)\lambda_\pi \\ \text{s. t.} \quad & \sum_{\pi=1}^{\Pi} (\boldsymbol{A}x_\pi)\lambda_\pi \geq \boldsymbol{d} \\ & \sum_{\pi=1}^{\Pi} \lambda_\pi = 1 \end{aligned}$$

Note that from equation (1.1), the bound given by v_D corresponds exactly to the optimal value of the problem $\min\{cx \mid \boldsymbol{A}x \geq d,\ x \in conv(X)\}$:

Theorem 1.4 ([32]). $v_D = \min\{cx \mid \boldsymbol{A}x \geq d,\ x \in conv(X)\}$

Corollary 1.1. *$v_D \geq v_{LR}$, where v_{LR} is the linear relaxation value of (P').*

If integrality enforcement constraint $\sum_{\pi=1}^{\Pi} \lambda_\pi x_\pi \in \mathbb{N}^{n_1} \times \mathbb{R}^{n_2}$ is added to the Dantzig-Wolfe master LP, then the resulting ILP is a *Dantzig-Wolfe reformulation* of (P'). If $n_2 = 0$ then the integrality enforcement constraints can be stated as $\lambda_\pi \in \mathbb{N}$, for each $\pi \in \{1,...,\Pi\}$.

Dantzig-Wolfe master problem (MP) has a large number of variables, and is classically solved by a column generation algorithm. The idea is to find the optimal solution to (MP) by considering only a subset of variables and iteratively adding useful variables to construct the optimal solution. The column generation algorithm is as follows. Consider the restricted master problem (RMP) which features only a subset $\Lambda \subset \{\lambda_1,...,\lambda_\Pi\}$ of the variables. Let μ^* be an optimal

dual solution of (RMP). Solve the *pricing* problem, often called *column generation subproblem*, (PP) $v_{PP} = \min\{(cx_\pi) - \mu^* a_\pi \mid 1 \leq \pi \leq \Pi\}$, where a_π is the π^{th} column of the constraint matrix of (DW). Quantity $(cx_\pi) - \mu^* a_\pi$ is called the *reduced-cost* of plan π. If $v_{PP} < 0$, variable λ_π, where π minimizes (PP), is added to (RMP) with its objective and constraint coefficients (cx_π, a_π), and the process iterates until no improving variable is found. Finiteness and correctness of column generation follows from the principles of the simplex algorithm. Although when Π is large, the pricing problem (PP) may seem difficult to handle, in many applications (PP) can be reformulated into a well-structured optimization problem. More thorough introductions to column generation can be found in [16, 58].

A *Branch & Price* algorithm for an integer linear program is a Branch & Bound where at each node, the lower bound on the optimal value of the associated subproblem is obtained via column generation [17].

1.2 The Unit Commitment Problem

Consider a discrete time horizon $\mathcal{T} = \{1, ..., T\}$, a demand for electric power D_t, $t \in \mathcal{T}$, a set $\mathcal{N}$ of n production units providing power. A production plan determines at which time each production unit is up and which quantity of power it produces. The demand is satisfied if at each time t, the total production is greater than or equal to the demand D_t. The *Unit Commitment Problem* (UCP) is to find a production plan satisfying the demand constraints as well as some operational constraints, while minimizing the total operating cost of each unit.

Note that in the literature, the demand constraint also appears as an equality constraint between the total production and the demand D_t.

1.2.1 Operational constraints and costs of the UCP

Costs The total operating cost is the sum of the operating cost $C^i(\pi)$ of each unit i, where π is the production plan followed by i. In the literature, $C^i(\pi)$ is decomposed as follows: $C^i(\pi) = C^i_0(\pi) + C^i_p(\pi)$. The start-up cost $C^i_0(p)$, incurred each time the unit starts up, is an exponential function of the unit down time, and the production cost $C^i_p(\pi)$ is a quadratic function of the quantity of power produced at each time period.

Main operating constraints At each time period, unit $i \in \mathcal{N}$ must be either down or up, and in the latter case, its production is within *production limits* $[P^i_{min}, P^i_{max}]$.

Each unit must satisfy *min-up* (resp. *min-down*) constraints, *i.e.* each unit i must remain up (resp. down) during at least L^i (resp. ℓ^i) periods after start up (resp. shut down). Without loss of generality, we consider that L^i, $\ell^i \leq T - 1$.

Operating constraints of each unit Each units i must satisfy *ramp-up* (resp. *ramp-down*) constraints, *i.e.*, the maximum increase (resp. decrease) in generated power from time period t to time period $t+1$ is RU^i (resp. RD^i). Moreover, *start-up* (resp. *shut down*) *ramp* constraints must be satisfied, *i.e.*, if unit i starts up at time t (resp. shuts down at time $t+1$), its production level at time t must be less than or equal to SU^i (resp. SD^i).

Some units feature *start-up* (resp. *shut-down*) *trajectories*. In such a case, the start-up (resp. shut-down) of unit i does not take only one time period but t_u^i (resp. t_d^i) time periods, during which the unit follows a specific power trajectory $P_1^{U,i}$, ..., $P_{t_u^i}^{U,i}$ (resp. $P_1^{D,i}$, ..., $P_{t_d^i}^{D,i}$).

At EDF, in addition to the previous constraints, the number of start-ups of the unit can be limited over given time spans. Moreover, the set $\mathscr{P}^i$ of feasible power outputs of unit i, $i \in \mathscr{N}$, is finite [18, 80]. Unit i is then said to have *finite-power-outputs*. In this case, once unit i reaches a stable production level, it must satisfy a *minimum operation time* constraint, *i.e.*, the unit's production must be constant or within a restricted range for a given time. A *modulation* is a change of stable production level. The maximum number of modulations of one unit over given time spans may be limited, typically for nuclear units.

Initial conditions At the beginning of the time horizon considered, each unit i is either up or down, and if the last start-up (resp. shut-down) occurred during the L^i (resp. ℓ^i) previous periods, then unit i still has to remain up (resp. down) until the min-up (resp. down) is satisfied.

Intra-site constraints Units located on the same production site share resources and must satisfy intra-site constraints [18]. Indeed, the unit set $\mathscr{N}$ is partitioned into K sites Σ_1, ..., Σ_K. The *intra-site* constraints are satisfied if at most one unit per site Σ_k, $k \in \{1,...,K\}$, starts up at each time period t.

Reserve requirements The European power system is a large interconnected system operating at uniform frequency. The system frequency must be maintained at its nominal level (50 Hz) in order to ensure a safe and optimal use of electrical equipment. To this end, the trade-off between power production and demand must be ensured in real-time, due to stochastic variations of the power demand (following unexpected weather conditions for example) and of the power production (following unexpected failure of a production unit for example). Therefore, electricity generation company must include power reserves in their production plan to be able to adjust the production to the demand at all times. Three reserve types, corresponding to distinct needs, must be provided: primary, secondary and tertiary reserves [81].

- *Primary reserve*, provided by the speed regulators of production units, quickly (within seconds) restores the demand/production equilibrium after a perturbation, if the available primary reserve is sufficient.

- Readjusting power production to the demand does not always restore the frequency to its nominal value. *Secondary reserve* allows the system to recover this nominal frequency. As each generation company provides primary reserve (even if not responsible for the perturbation), primary reserve supply unbalanced power exchanges between companies. Secondary reserve brings exchanges back to their contractual values.

- *Tertiary reserve* helps rebuild low secondary reserves, in order to anticipate any new perturbation. It is decomposed with respect to activation times.

At EDF, primary and secondary reserves are treated as demand constraints, while tertiary reserve is not included in UCP formulations. Other definitions of power reserves can be found in the literature.

1.2.2 Resolution of the UCP by Lagrangian relaxation

Even though the UCP has been subject to a large research activity (see survey [82], its update [74] and more recently [84, 88]), it still cannot be regarded as a well-solved problem.

Historically, the UCP has been solved by Lagrangian relaxation, in the literature [68] as well as in the industry.

EDF (Électricité de France) manages a mix of production units composed of nearly 60 nuclear power plants, 20 fossil-fuel power plants (3 coal-fired power plants, 13 fuel oil or gas turbines and 4 gas combined cycle power plants), and 500 hydropower plants dispatched in 50 valleys.

Fuel oil and gas turbines are highly manoeuvrable, with small min-up/down times, and thus play a security role in the power system. Nuclear units have higher min-up/down times, but their production costs are competitive. Therefore they provide an important share of the total production when available. On this basis, units such as hydropower plants as well as gas combined cycle or coal power plants can be used to fit exactly the total production to the demand.

Given the large number and variety of units, the daily production planning problem is solved at EDF [18, 80] using a Lagrangian relaxation [22] – commonly referred to as price decomposition – where the coupling demand constraints are dualized and the prices are updated using a subgradient method. Each nuclear or fossil-fuel unit (resp. each valley) is treated as a subproblem and is solved using dynamic programming (resp. linear programming). A classical Lagrangian decomposition is performed at a first stage, where the subproblems must be solved exactly in order to ensure the convergence of the decomposition scheme. Thus, some constraints of these subproblems are relaxed in practice. For example, integrality constraints in subproblems corresponding to hydro valleys are dropped. Similarly, intra-site constraints for fossil-fuel units are not modeled. This Lagrangian relaxation does not systematically produce feasible solutions, therefore an augmented Lagrangian relaxation [18] is considered at a second stage in order to improve feasibility recovery.

Several techniques have been proposed in the literature to solve the UCP (see surveys [84, 88]). A large part of the literature [5, 68] deals with decomposition schemes whose subproblems featuring non-linearity and non-concavity are solved using dynamic programming. When no ramp constraints are considered, subproblems corresponding to one unit can be polynomially solved using classical dynamic programming schemes where a state (s,d) indicates that the unit has been in up/down state s for d time periods [5]. Polynomial dynamic programming schemes have also been proposed for the UCP with one production unit featuring min-up/min-down and ramp constraints alongside with down time dependent start-up costs [20, 24, 36]. The state space in [20] is represented as a network where state (d, s, k) indicates that the unit switches to up/down status s for the k^{th} time, and the unit remains in status s for a duration d. The transition cost is obtained by solving an optimal production dispatch problem including ramp constraints. In [24], a similar scheme is proposed, alongside with an algorithm to solve the production dispatch problem with arbitrary convex cost functions. Variants are studied in [36].

1.2.3 The Min-up/min-down Unit Commitment Problem

The *Min-up/min-down Unit Commitment Problem* (MUCP) is to find a production plan minimizing the total operating cost while satisfying production limits, demand and min-up/down time constraints. The total operating cost is the sum of the operating cost $C^i(\pi)$ of each unit i, defined as

$$C^i(\pi) = \sum_{t=1}^{T} c_p^i p(\pi,i,t) + c_f^i \,\mathrm{up}(\pi,i,t) + c_0^i \,\mathrm{startup}(\pi,i,t)$$

where c_p^i, c_f^i, $c_0^i \in \mathbb{R}$, $p(\pi,i,t)$ is the quantity of power produced by unit i at time t in production plan π, $\mathrm{up}(\pi,i,t)$ equals 1 if unit i is up at time t in production plan π, and 0 otherwise, and $\mathrm{startup}(\pi,i,t)$ equals 1 if unit i is starts up at time t in production plan π, and 0 otherwise.

Example 1.1. *Consider an illustrative instance of the MUCP with $T = 3$, $D = [20, 10, 25]$ and three units having the characteristics given in Table 1.1.*

	P^i_{min}	P^i_{max}	ℓ^i	L^i	c^i_f	c^i_0	c^i_p
Unit 1	5	15	2	2	10	10	2
Unit 2	5	5	2	2	5	5	10
Unit 3	5	5	2	2	5	5	10

Table 1.1: Characteristics of the units of Example 1.1

Figure 1.1 represents the demand with dotted lines, and illustrates the solution in which unit 1 and 2 are up at all times, and unit 3 is down at times 1 and 2 and up at time 3. This solution has a total cost of 255. Note that unit 1 alone can produce enough to satisfy the demand at time 2. It would cost less to shut down unit 2 at time 2 and start it up again at time 2, but doing so would not respect the min-down time $\ell^2 = 2$ of unit 2.

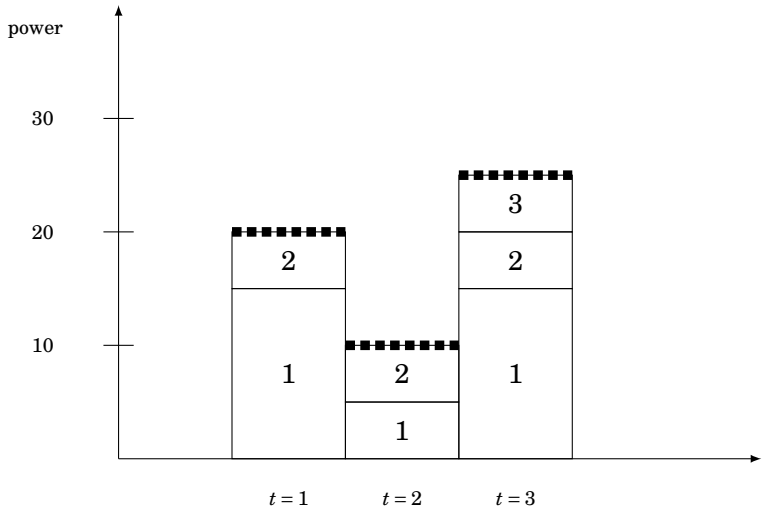


Figure 1.1: Demand values and production plan relative to Example 1.1

From a combinatorial point of view, the Min-up/min-down Unit Commitment Problem (MUCP) is the core structure of the thermal UCP solved daily at EDF.

In order to study the combinatorial aspects of the UCP, this thesis will focus on the MUCP with possibly additional constraints coming from the UCP, such as ramp or intra-site constraints.

In perspective of a decomposition scheme for the UCP, a Lagrangian relaxation is commonly used. Classically, the only dualized coupling inequalities are the demand and reserve constraints. The other coupling constraints are then either left in the subproblems, as ramp-constraints, or unmodeled, as the intra-site constraints.

The Intra-site Min-up/min-down Unit Commitment Problem (IMUCP) is a generalization of the MUCP where the intra-site constraints must be satisfied. In a decomposition scheme for the IMUCP, the demand constraint is still dualized while the intra-site constraints can remain in the pricing problem. The pricing problem is then divided into k subproblems, one for each site. Each subproblem contains all the IMUCP constraints, but the demand satisfaction. To take into account the demand constraint, the fixed and proportional production costs are modified, leading to the so-called *reduced costs*. For each unit i and time t, the reduced cost related to the fixed cost of unit i is denoted by $\pi^{i,t}$ and that of proportional production cost by $\rho^{i,t}$. In this case, these two costs can be negative and depend on t and i. The pricing subproblem of the IMUCP is called P-IMUCP. Note that this problem also arises as the pricing subproblem in a column generation setting.

The MUCP variants considered in this thesis are thus the following

- the MUCP,
- the ramp-constrained MUCP,
- the intra-site MUCP (IMUCP)
- the pricing subproblem of the IMUCP (P-IMUCP).

1.2.4 Instances

In order to account for the diversity of combinatorial issues arising in the MUCP and its variants, we consider multiple types of instances.

Preliminary experimental results indicate that the tightness of the production range $[P^i_{min}, P^i_{max}]$ deeply impacts the difficulty of MUCP instances. In the dataset presented in [13], P^i_{min} is around 25 % of P^i_{max} for each unit i. On the opposite, for realistic units at EDF, the tightness, *i.e.*, the ratio $\frac{P^i_{min}}{P^i_{max}}$, varies from to 25 to 70% depending on the unit's type. Moreover, our preliminary results show that when P^i_{min} is close to P^i_{max} (by 75% or more), then MUCP instances become very difficult. Therefore we consider three classes of instances, namely literature, realistic and tight-production-range) taking these differences into account.

Moreover, it is well known that symmetries in the MUCP also strongly affect the computation time. Thus, we consider instances with symmetries, by duplicating production units, and instances without symmetries.

In the case of the IMUCP, it is likely that the demand profile over the time horizon changes the impact of intra-site constraints, as huge demand variations may require to start-up multiple units at the same time. Therefore, we consider two possible demand profiles: the classical 2-peak-per-day profile, and a random demand profile.

The instances used for computational experiments are generated as follows.

Instance classes We consider the following classes of instances:

- R: The realistic (R) instances are generated using data for real EDF units. We partition the units into three types, depending on their fuel: coal, gas and fuel oil. For each fuel type, we consider the characteristics (P_{min}, P_{max}, L, ℓ, c_f, c_0, c_p) of each real EDF unit, and we draw a correlation matrix between their characteristics. Moreover there is a typical range for each characteristic depending on the fuel. Thus, for each instance, we generate $\frac{n}{3}$ units with the characteristics based on the correlations and ranges of each fuel.

- L: The literature (L) instances are similarly generated, using the unit characteristics from the dataset presented in [13]. Note that in this class only one type of unit is considered, as the units characteristics appear to be similar to each other in the dataset from [13].

The tightness of the production range $[P^i_{min}, P^i_{max}]$ for each unit i deeply affects the computation time of the MUCP. This observation leads us to generate another instance class.

- TPR: The *tight-production-range* (TPR) instances are generated as literature instances in which, for each unit i, we set P^i_{min} as a percent of P^i_{max}, namely 50 %, 75 % and 100 %. These classes are respectively denoted by TPR-50, TPR-75 and TPR-100. In these instances, P_{min} is closer to P_{max} than in the first two classes. As a basis for comparison, in the literature class, P_{min} is around 25 % of P_{max}, and varies from to 25 to 70% in the realistic class. Note that operating rules applying to the practical UCP lead to restrict the unit

production range considered in the MUCP. Recall from Section 1.2.1 that thermal units must satisfy ramp constraints restricting the power output variation at each time period. Moreover, some EDF units have finite-power-outputs (see Section 1.2.1). Ramp-constrained units could be approximated by MUCP units with a tight (for example 75%) production range. Similarly, finite-power-outputs units could be approximated by MUCP units with a 100% tight production range. Therefore, TPR instances are designed to give us an insight into the potential effectiveness of our algorithms for the real-world UCP.

Symmetries In the dataset from [13], symmetries are introduced by duplicating production units. We thus generate instances with symmetries (S) and instances without symmetries (NS) for each class R, L, TPR-50, TPR-75 and TPR-100. Units of NS instances are randomly generated according to the procedures previously described. Units of S instances are generated as follows: some units are randomly generated and then are duplicated d times, where d is randomly selected in $[1, \frac{n}{F}]$ for each unit, in order to obtain a total of n units. Parameter $F \in \mathbb{N}$ is called the symmetry factor.

Demand constraints We consider two demand profiles:

- 2-peak-demand instances: we generate a "2-peak per day" type demand with a large variation between peak and off-peak values: during one day, the typical demand in energy during one day has two peak periods, one in the morning and one in the evening. The amplitudes between peak and off-peak periods have similar characteristics to those in the dataset from [13].
- Random-demand instances: at each time t, the demand is randomly generated, according to a uniform distribution from 0 to $\sum_{i \in \mathcal{N}} P^i_{max}$.

Intra-site constraints In order to define unit sites, we first select a site size s at random in $[1,6]$. Then s units are randomly generated to form a site. This process is repeated until n units are obtained.

Ramp constraints The ramp-constrained MUCP instances considered are the same as in the non-ramp-constrained case, with additional ramp characteristics $RU^j = \frac{P^j_{max} - P^j_{min}}{3}$, $RD^j = \frac{P^j_{max} - P^j_{min}}{2}$ and $SU^j = SD^j = P^j_{min}$.

1.2.5 ILP formulations of the Min-up/min-down Unit Commitment Problem

Various ILP formulations for the MUCP are given in this section. Each of these formulations induce an integral polytope when $n = 1$, *i.e.*, when only one production unit is considered. When additional technical constraints are taken into account (see Section 1.2.1), the integrality property may not hold anymore depending on the formulation considered.

• (x,u) **formulation** $(F^n_{x,u})$ This formulation is based on the work of Rajan and Takriti [79].

For each unit $i \in \mathcal{N}$ and time period $t \in \mathcal{T}$, variable $x^i_t \in \{0,1\}$ equals 1 if and only if unit i is up at time t, and variable $p^i_t \in \mathbb{R}$ represents the quantity of power produced by unit i at time t. For each unit $i \in \mathcal{N}$ and time period $t \in \{2,...,T\}$, variable $u^i_t \in \{0,1\}$ equals 1 if and only if unit i starts up at time t. Formulation $(F^n_{x,u})$ for the MUCP is as follows:

$$
\begin{aligned}
(F^n_{x,u}) \quad \min_{x,u,p} \quad & \sum_{i=1}^{n}\sum_{t=1}^{T} c^i_f x^i_t + c^i_p p^i_t + c^i_0 u^i_t & & \\
\text{s. t.} \quad & \sum_{t'=t-L^i+1}^{t} u^i_{t'} \le x^i_t & \forall i \in \mathcal{N},\ \forall t \in \{L^i+1,...,T\} & \quad (1.2)\\
& \sum_{t'=t-\ell^i+1}^{t} u^i_{t'} \le 1 - x^i_{t-\ell^i} & \forall i \in \mathcal{N},\ \forall t \in \{\ell^i+1,...,T\} & \quad (1.3)\\
& u^i_t \ge x^i_t - x^i_{t-1} & \forall i \in \mathcal{N},\ \forall t \in \{2,...,T\} & \quad (1.4)\\
& \sum_{i=1}^{n} p^i_t \ge D_t & \forall t \in \mathcal{T} & \quad (1.5)\\
& P^i_{min} x^i_t \le p^i_t \le P^i_{max} x^i_t & \forall i \in \mathcal{N},\ \forall t \in \mathcal{T} & \quad (1.6)\\
& x^i_t \in \{0,1\} & \forall i \in \mathcal{N},\ \forall t \in \mathcal{T} & \quad (1.7)\\
& u^i_t \in \{0,1\} & \forall i \in \mathcal{N},\ \forall t \in \{2,...,T\} & \quad (1.8)
\end{aligned}
$$

The set of all feasible $x = (x^i_t)_{t\in\mathcal{T}, i\in\mathcal{N}}$ is denoted by $\mathcal{X}_{MUCP}$.

Inequalities (1.2), (1.3) and (1.4), are introduced in [79]. Inequality (1.2) is the minimum up-time constraint: it states that if unit i is down at time t, then it cannot have started up during the L^i previous periods. Inequality (1.3) is the minimum down-time constraint, which is symmetric to the minimum up-time constraint. Inequality (1.4) ensures that if unit i starts up at time t (i.e. $x^i_t - x^i_{t-1} = 1$) then start-up variable u^i_t must equal 1. Inequality (1.6) sets bounds to the quantity of power produced by each unit, and inequality (1.5) ensures that the demand is satisfied at each time period.

Theorem 1.5, relative to the 1-unit case, is proved in [79].

Theorem 1.5 ([59, 79])**.** *The following polytope is integral*

$$P^1_{x,u} = \left\{ x^1_t,\, u^1_t \in [0,1] \mid t \in \{1,...,T\},\ (1.2),\ (1.3),\ (1.4) \right\}.$$

This result has been proved independently by Malkin [59], who shows that the corresponding constraint matrix is totally unimodular.

When ramp-constraints are taken into account, variables ρ replace variables p where ρ^i_t is defined for each $i \in \mathcal{N}$ and $t \in \mathcal{T}$ as

$$
\begin{aligned}
\rho^i_t &= p^i_t - P^i_{min} && \text{if } x^i_t = 1\\
\rho^i_t &= 0 && \text{otherwise}
\end{aligned}
$$

Constraints (1.6) and (1.5) become

$$0 \leq \rho_t^i \leq (P_{max}^i - P_{min}^i)x_t^i \quad \forall i \in \mathcal{N},\ \forall t \in \{1,...T\} \tag{1.9}$$

$$\sum_{i=1}^{n} \left(P_{min}^i x_t^i + \rho_t^i\right) \geq D_t \qquad \forall t \in \{1,...T\} \tag{1.10}$$

Using x, u and ρ variables, *ramp constraints* can be formulated as follows:

$$\rho_t^i - \rho_{t-1}^i \leq RU^i x_{t-1}^i + (SU^i - P_{min}^i)u_t^i \quad \forall i \in \mathcal{N},\ \forall t \in \{2,...T\} \tag{1.11}$$

$$\rho_{t-1}^i - \rho_t^i \leq RD^i x_t^i + (SD^i - P_{min}^i)w_t^i \qquad \forall i \in \mathcal{N},\ \forall t \in \{2,...T\} \tag{1.12}$$

Using x, u and p variables, *ramp constraints* would be formulated as follows:

$$p_t^i - p_{t-1}^i \leq RU^i x_{t-1}^i + SU^i u_t^i \quad \forall i \in \mathcal{N},\ \forall t \in \{2,...T\} \tag{1.13}$$

$$p_{t-1}^i - p_t^i \leq RD^i x_t^i + SD^i w_t^i \qquad \forall i \in \mathcal{N},\ \forall t \in \{2,...T\} \tag{1.14}$$

Note that inequalities (1.11) and (1.12) are tighter than inequalities (1.13) and (1.14) in the sense that some fractional solutions (x,u,p) would not exist in the (x,u,ρ) space. For example, suppose $n=1$, $T=2$, $SU^1 = P_{min}^1$ and $x^1 = [0.5,\ 1]$. Then the production plan $p^1 = [\frac{P_{max}^1}{2},\ \frac{P_{max}^1}{2} + \frac{RU^1}{2} + \frac{P_{min}^1}{2}]$ is feasible for the (x,u) formulation featuring variables p. On the opposite, any solution (x,u,ρ) is such that $\rho_2^1 \leq \frac{P_{max}^1}{2} + \frac{RU^1}{2}$, by ramp-up constraint (1.11).

In the non-ramp-constrained MUCP case, using p or ρ variables does not change the linear relaxation value.

• **Flow formulation** (***F^n-Flow***) A flow formulation for the sequences of start-ups and shut-downs of each unit is introduced in [78]. Consider a unit $i \in \mathcal{N}$ and two time periods $t,t' \in \{1,...,T+1\}$. Basically, variable $f^i(t,t')$ (resp. $g^i(t,t')$) equals 1 if unit i starts up (resp. shuts down) at time t, remains up (resp. down) from t to $t'-1$ and shuts down (resp. starts up) at time t, satisfying min-up (resp. min-down) times. More formally,

- For $t \in \{2,...,T\}$ and $t' \geq t + L^i$ (resp. $t' \geq t + \ell^i$), variable $f^i(t,t')$ (resp. $g^i(t,t')$) $\in \{0,1\}$ equals 1 if and only if unit i starts up (resp. shuts down) at time t, remains up (resp. down) until time $t'-1$ and shuts down (resp. starts up) at time t'.

- For $t \in \{2,...,T\}$ and $t' = T+1$, variable $f^i(t,t')$ (resp. $g^i(t,t')$) $\in \{0,1\}$ equals 1 if and only if unit i starts up (resp. shuts down) at time t and remains up (resp. down) from time t to time T.

- For $t = 1$ and $t' \in \{2,...,T\}$, variable $f^i(t,t')$ (resp. $g^i(t,t')$) $\in \{0,1\}$ equals 1 if and only if unit i is up (resp. down) from time 1 to time $t'-1$ and shuts down (resp. starts) up at time t'.

- For $t = 1$ and $t' = T+1$, variable $f^i(t,t')$ (resp. $g^i(t,t')$) $\in \{0,1\}$ equals 1 if and only if unit i is up (resp. down) from time 1 to time T.

All other variables $f^i(t,t')$ and $g^i(t,t')$ are 0. Flow formulation (F^n-*Flow*) is as follows.

$$
\begin{aligned}
(F^n\text{-}Flow) \quad \min_{f,g,p} \quad & \sum_{i=1}^{n}\left(\sum_{t=1}^{T}\sum_{t'=2}^{T+1} c^i_{t,t'} f^i(t,t') + \sum_{t=1}^{T} c^i_p p^i_t\right) & & \\
\text{s. c.} \quad & \sum_{t'=1}^{t-1} g^i(t',t) - \sum_{t'=t+1}^{T+1} f^i(t,t') = 0 & \forall i \in \mathcal{N},\ \forall t \in \{1,...,T\} & (1.15)\\
& \sum_{t'=1}^{t-1} f^i(t',t) - \sum_{t'=t+1}^{T+1} g^i(t,t') = 0 & \forall i \in \mathcal{N},\ \forall t \in \{1,...,T\} & (1.16)\\
& \sum_{t'=2}^{T+1} f^i(1,t') + g^i(1,t') = 1 & \forall i \in \mathcal{N} & (1.17)\\
& P^i_{min}\left(\sum_{t'=1}^{t}\sum_{t''=t+1}^{T+1} f^i(t',t'')\right) \le p^i_t & \forall i \in \mathcal{N},\ t \in \{1,...T\} & (1.18)\\
& p^i_t \le P^i_{max}\left(\sum_{t'=1}^{t}\sum_{t''=t+1}^{T+1} f^i(t',t'')\right) & \forall i \in \mathcal{N},\ \forall t \in \{1,...T\} & (1.19)\\
& \sum_{i=1}^{n} p^i_t \ge D_t & \forall t \in \{1,...T\} & \\
& f^i(t,t'),\ g^i(t,t') \in \{0,1\} & \forall i \in \mathcal{N},\ \forall t,\ t' \in \{1,...,T+1\} &
\end{aligned}
$$

where the cost associated to $f^i(1,t')$ is $c^i_{1,t'} = (t'-1)c^i_f$, and for $t \ge 2$, the cost associated to $f^i(t,t')$ is $c^i_{t,t'} = (t'-t)c^i_f + c_0$.

For a given unit i, consider a bipartite graph G with vertices $V = V^U \cup V^D \cup \{p\}$, where for each $t \in \mathcal{T}$, $v^u_t \in V^U$ corresponds to a start-up of unit i at time t, $v^d_t \in V^D$ corresponds to a shut down at time t and p is a sink node. For each t, $t' \in \mathcal{T}$, the arc associated with flow variable $f^i(t,t')$ (resp. $g^i(t,t')$) connects the start-up (resp. shut down) at time t to the shut-down (resp. start-up) at time t'. If $t' \ge t + L^i$ (resp. $t' \ge t + \ell^i$) this sequence is feasible. The arc associated with flow variable $f^i(t,T+1)$ (resp. $g^i(t,T+1)$) connects the start-up (resp. shut down) at time t to the sink node.

In the one-unit case, *i.e.*, $n = 1$, and when $D_t = 0$ for each $t \in \mathcal{T}$, formulation F^n(-*Flow*) minimizes the cost of a unitary flow in graph G. This unitary flow corresponds to a feasible up/down plan for unit i. In this case the constraint matrix is the arc-vertex incidence matrix of G. As the incidence matrix of an oriented graph is totally unimodular [64], the following polytope is integral

$$P^1_{flow} = \left\{ f^1(t,t'),\ g^1(t,t') \ge 0 \mid t,\ t' \in \{1,...,T+1\},\ (1.15),\ (1.16),\ (1.17) \right\}.$$

Example 1.2. *Consider an MUCP instance with $T = 4$ and $n = 1$ unit, with min-up time $L^1 = 2$ and min-down time $\ell^1 = 3$. Figure 1.2 presents a solution to formulation (F^1-Flow). The bold arcs in bipartite graph G represent a feasible up/down plan for the unit: the unit is down at time 1, starts up at time 2, remains up at time 3 and shuts down at time 4.*

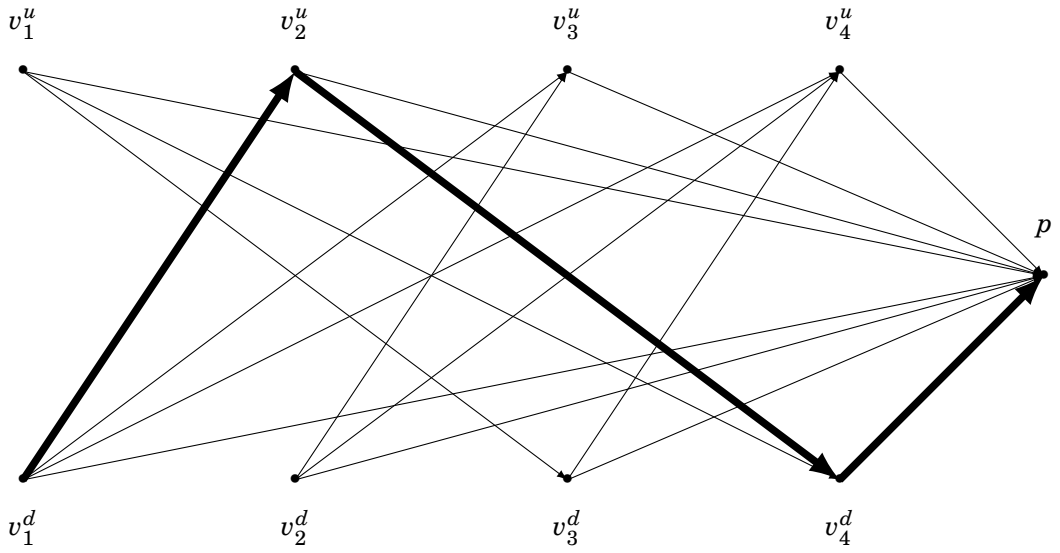


Figure 1.2: A solution to formulation (F^n-*Flow*) with $n = 1$ and $T = 4$

• **Interval formulation** (F^n-***Int***) A variant of the flow formulation is the interval formulation introduced in [49]. For each unit $i \in \mathcal{N}$, for each interval $\{t_0, ..., t_1 - 1\}$ of size $t_1 - t_0 \geq L^j$, variable $y^i(t_0, t_1)$ equals 1 if and only if unit i starts up at time t_0, remains up on interval $\{t_0, ..., t_1 - 1\}$ and shuts down at time t_1. For each time period $t \in \mathcal{T}$, variable $p_t^i(t_0, t_1)$ represents the quantity of power produced by unit i at time t if $y^i(t_0, t_1) = 1$, and $p_t^i(t_0, t_1) = 0$ otherwise. The formulation is as follows.

$$(F^n\text{-}Int) \quad \min_{y,p} \sum_{j=1}^{n} \sum_{\{t_0,...,t_1-1\} \in \mathscr{Y}_j} c^i(t_0, t_1) y^i(t_0, t_1) + c_p^j \sum_{t=t_0}^{t_1-1} p_{t,j}^{t_0,t_1}$$

$$\text{s. t.} \qquad A^i(t_0, t_1) p_t^i(t_0, t_1) \leq b^i(t_0, t_1) y^i(t_0, t_1) \qquad \forall i \in \mathcal{N},\ \forall \{t_0, ..., t_1 - 1\} \in \mathscr{Y}_i \quad (1.20)$$

$$\sum_{\substack{\{t_0,...,t_1-1\} \in \mathscr{Y}_i \\ \text{s.t. } t \in \{t_0,...,t_1+\ell^i\}}} y^i(t_0, t_1) \leq 1 \qquad \forall i \in \mathcal{N},\ \forall t \in \mathcal{T} \quad (1.21)$$

$$\sum_{j \in \mathcal{N}} \sum_{\{t_0,...,t_1-1\} \in \mathscr{Y}_j} p_t^i(t_0, t_1) \geq D_t \qquad \forall t \in \mathcal{T} \quad (1.22)$$

$$y_t^i(t_0, t_1) \in \{0, 1\} \qquad \forall j \in \mathcal{N},\ \forall \{t_0, ..., t_1 - 1\} \in \mathscr{Y}_i \quad (1.23)$$

where $\mathscr{Y}_i = \{\{t_0, ..., t_1 - 1\} \in T \times T \mid t_1 - t_0 \geq L^i\}$ and where

$$\mathscr{P}^i(t_0, t_1) = \{p^i(t_0, t_1) \in \mathbb{R}_+^T \mid A^i(t_0, t_1) p^i(t_0, t_1) j \leq b^i(t_0, t_1)\}$$

is the feasible production polytope of unit i, if unit i starts up at time t_0, remains up on interval $\{t_0, ..., t_1 - 1\}$ and shuts down at time t_1. In the MUCP case, this polytope is defined by production

limits (1.24).

$$P^i_{min} \le p^i_t(t_0,t_1) \le P^i_{max} \tag{1.24}$$

In the one-unit case, *i.e.*, $n = 1$, and when $D_t = 0$ for each $t \in \mathcal{T}$, formulation (F^n-*Int*) optimizes over stable sets in an interval graph G. Graph G is such that where each vertex corresponds to a feasible interval $\{t_0,...,t_1+\ell^i-1\}$ such that the unit is up on $\{t_0,...,t_1-1\}$ and down on $\{t_1,...t_1+\ell^i-1\}$. Thus a stable set in G corresponds to a feasible up/down plan for the unit. As the convex hull of the stable set problem is completely described by clique and nonnegativity inequalities for interval graphs [34], the following polytope is integral

$$P^1_{Int} = \Big\{ y^1(t_0,t_1) \ge 0 \mid \{t_0,...,t_1-1\} \in \mathscr{Y}_i, \text{ (1.21)} \Big\}.$$

• **Aggregated demand-coupling formulations** Consider a *demand-coupling* formulation for the n-unit MUCP, *i.e.*, a formulation such that the only coupling inequalities are demand constraints:

$$\begin{array}{lll} (F^n_{dc}) \quad \min\limits_{z,p} & \boldsymbol{c_z z} + \boldsymbol{c_p p} & \\ \text{s. t.} & (z^i,p^i) \in \Pi^i_F & \forall i \in \mathscr{N} \\ & \sum\limits_{i=1}^{n} p^i_t \ge D_t & \forall t \in \mathscr{T} \\ & z^i \in \mathbb{Z}^m,\ p^i \in \mathbb{R}^{(n,T)} & \forall i \in \mathscr{N} \end{array}$$

where (c_z,c_p) is the cost vector and $\Pi^i_F = \{A^i z^i + B^i p^i \le d^i\}$ is a polyhedron such that $\Pi^i_F \cap (\mathbb{Z}^m \times \mathbb{R}^{(n,T)})$ is a set of feasible plans for unit i, expressed with arbitrary variables $z^i \in \mathbb{Z}^m$ and production variables p^i_t.

Suppose the units can be partitioned into H sets $\mathscr{N} = \mathscr{N}_1 \cup ... \cup \mathscr{N}_H$ such that for each $h \in \{1,...,H\}$, for all $i,\ j \in \mathscr{N}_h$, $A^i = A^j = \mathscr{A}^h$, $B^i = B^j = \mathscr{B}^h$, $d^i = d^j = \overline{d}^h$, $c^i_z = c^z_j = \overline{c}^h_z$ and $c^i_p = c^p_j = \overline{c}^h_p$.

Linear inequality system $\mathscr{A}^h z + \mathscr{B}^h p \le \overline{d}^h$ has the integer decomposition property (see Theorem 1.3) if for any integer k and $(\overline{z},\overline{p}) \in \mathbb{Z}^n \times \mathbb{R}^{(n,T)}$ such that $\mathscr{A}^h \overline{z} + \mathscr{B}^h \overline{p} \le k\overline{d}^h$, there exist z_1, ..., $z_k \in \mathbb{Z}^n$, p_1, ..., $p_k \in \mathbb{R}^{(n,T)}$ such that $\mathscr{A}^h z_{k'} + \mathscr{B}^h p_{k'} \le \overline{d}^h$, $k' \in \{1,...,k\}$, and $\overline{z} = z_1 + ... + z_k$, $\overline{p} = p_1 + ... + p_k$.

If this property holds, then variables (z^i,p^i), $i \in \mathscr{N}_h$ can be aggregated into variables $(\overline{z}^h,\overline{p}^h) = \sum_{i\in\mathscr{N}_h}(z^i,p^i)$, $h \in \{1,...,H\}$, resulting in the following aggregated formulation

$$\begin{array}{lll} (A-F^n_{dc}) \quad \min\limits_{\overline{z},\overline{p}} & \boldsymbol{\overline{c}^h_z z} + \boldsymbol{\overline{c}^h_p p} & \\ \text{s. t.} & \overline{A}^h \overline{z}^h + \overline{B}^h \overline{p}^h \le |\mathscr{N}_h| \overline{d}^h & \forall h \in \{1,...,H\} \\ & \sum\limits_{h=1}^{H} \overline{p}^h_t \ge D_t & \forall t \in \mathscr{T} \\ & \overline{z}^h \in \mathbb{Z}^m & \forall h \in \{1,...,H\} \end{array}$$

The integer decomposition property ensures that an integer aggregated solution $(\overline{z}^h, \overline{p}^h)$, $h \in \{1,...,H\}$, can be disaggregate into $|\mathcal{N}_h|$ integer solutions $(z^i, p^i) \in \Pi_F^i \cap (\mathbb{Z}^m \times \mathbb{R})$.

Aggregated (x,u) and interval formulations, introduced in [50], are detailed in Section 5.4.

Note that formulations $(F^n_{x,u})$, $(F^n\text{-}Flow)$ and $(F^n\text{-}Int)$ are demand-coupling formulations.

• **Dantzig-Wolfe reformulations** Consider an MILP formulation of the MUCP:

$$\begin{array}{lll} (F) & \min\limits_{v} & \boldsymbol{cv} + \boldsymbol{dp} \\ & \text{s. t.} & Av + Bp \leq b \\ & & p \in \mathbb{R}^q_+ \\ & & v \in X \end{array}$$

where $c \in \mathbb{R}^m$, $d \in \mathbb{R}^q$, A and B are matrices and $X \subseteq \mathbb{Z}^{m_1} \times \mathbb{R}^{m_2}$ is a bounded mixed integer set.

Let P be the set of extreme points of X. Then for each $v \in X$, there exists $\lambda_\pi \geq 0$, $\pi \in P$, such that $\sum_{\pi \in P} \lambda_\pi = 1$ and $v = \sum_{\pi \in P} \lambda_\pi \pi$.

Then the Dantzig-Wolfe reformulation of (F) is

$$\begin{array}{lll} (DW) & \min\limits_{v} & \sum\limits_{\boldsymbol{\pi \in P}} \boldsymbol{c_\pi \lambda_\pi} + \boldsymbol{dp} \\ & \text{s. t.} & \sum\limits_{\pi \in P} a_\pi \lambda_\pi + Bp \leq b \\ & & \sum\limits_{\pi \in P} \lambda_\pi = 1 \\ & & p \in \mathbb{R}^q_+ \\ & & \lambda_\pi \geq 0 \qquad\qquad \forall \pi \in P \\ & & \sum\limits_{\pi \in P} \lambda_\pi \pi \in \mathbb{Z}^{m_1} \times \mathbb{R}^{m_2} \end{array}$$

where $a_\pi = A\pi$ and $c_\pi = c\pi$, for each $\pi \in P$.

Referring to the framework presented in Section 1.1.3, $Av + Bp \leq b$ are the dualized constraints, and the constraints of the column generation subproblem are satisfied by each $v \in X$.

Note that if $X \subseteq \mathbb{Z}^{m_1}$ is an integer set, then integrality enforcement constraint $\sum_{\pi \in P} \lambda_\pi \pi \in \mathbb{Z}^{m_1}$ can be replaced by $\lambda_\pi \in \{0,1\}$, for each $\pi \in P$.

1.2.6 Polyhedral studies of the 1-unit UCP

Several articles propose polyhedral studies for UCP variants with only one production unit. The min-up/min-down polytope $P^i(L^i, \ell^i)$ is introduced in [51]:

$$
\begin{aligned}
P^i(L^i,\ell^i) = \quad & \Big\{ x \in \{0,1\}^T \text{ s.t. } \forall t \in \{2,...,T\} \\
& x^i_t - x^i_{t-1} \le x_\tau && \forall \tau \in \{t+1,...,\min(t+L^i,T)\} && (1.25)\\
& x^i_{t-1} - x^i_t \le 1 - x^i_\tau && \forall \tau \in \{t+1,...,\min(t+\ell^i,T)\} && (1.26)\\
& \Big\}
\end{aligned}
$$

Inequalities (1.25) and (1.26), introduced in [85], enforce the minimum up and down time constraints of a single unit $i \in \mathcal{N}$. The authors of [51] give a complete linear description of $conv(P^i(L^i,\ell^i))$. Consider an integer $k \ge 0$ and integers $\phi(1)$, ..., $\phi(k+1)$, $\psi(1)$, ..., $\psi(k) \in \{1,...,T\}$, such that

$$\phi(1) < \psi(1) < \phi(2) < \psi(2) < ... < \phi(k) < \psi(k) < \phi(k+1)$$

If $\phi(k+1) - \phi(1) \le L$, the *alternating-up* inequality, introduced in [51], is defined as follows

$$-\sum_{j=1}^{k+1} x_{\phi(j)} + \sum_{j=1}^{k} x_{\psi(j)} \le 0. \tag{1.27}$$

Similarly, if $\phi(k+1) - \phi(1) \le \ell$, the *alternating-down* inequality is as follows

$$\sum_{j=1}^{k+1} x_{\phi(j)} - \sum_{j=1}^{k} x_{\psi(j)} \le 1. \tag{1.28}$$

It is shown in [51] that these inequalities completely describe $Conv(P^i(L^i,\ell^i))$. An exact polynomial-time separation algorithm is also devised. It follows that the corresponding problem can be solved in polynomial time for any cost values.

In case the unit has a start-up cost, additional binary variables u_t are needed to indicate whether the unit starts up at time t. In [79], the authors study the 1-unit polytope associated to the min-up and min-down constraints in the (x, u) variable space. They prove that inequalities (1.2), (1.3) and (1.4) completely describe this polytope. The authors of [31] extend this result, as they completely describe the polytope of the 1-unit problem with min-up/down constraints, production limits and start-up/shut-down ramp constraints. The polytope of the same problem with additional start-up and shut-down trajectories is completely described in [65].

When only production limits and ramp constraints are considered (note that there is no min-up/down times), the corresponding 1-unit polytope in (x,u,p) space is completely described for $T = 2$ in [14]. The linear inequalities of the description are valid for any T, and exponential classes of inequalities involving more than two time periods are introduced, with polynomial time separation. Facet conditions are also given.

When production limits, min-up/down and ramp constraints are considered, no complete description of the 1-unit polytope $\mathscr{P}^1_{ramp}$ in the (x,u,p) or (x,u,ρ) space is known. A tighter reformulation of production limits (1.6) in the (x,u,p) space taking advantage of ramp constraints (1.13) (1.14) is proposed in [71]

$$p^i_t \leq P^i_{max} x^i_{t+K(t)} + \sum_{k=1}^{K^i(t)} \left(SD^i + (k-1)RD^i\right) w^i_{t+k} - \sum_{k=1}^{K^i(t)} P_{max} u^i_{t+k},\ \forall t \in \mathscr{T},\ \forall i \in \mathscr{N}$$

where $K^i(t) = \max\left\{k \in \{1,...,L^i\} \text{ s.t. } SD^i + (k-1)RD^i \leq P^i_{max} \text{ and } k+t \leq |T|\right\}$.

Indeed, if unit i is up at time t, either it remains up on the interval $\{t,...,K^i(t)\}$, and then $p^i_t \leq P_{max}$, or it shuts down at time $t+t'$, $t' \leq K(t)$, and then $p_t \leq SD^i + (t'-1)RD^i$ is order to satisfy ramp constraints. If unit i is down at t then $p^i_t = 0$. In this case, if unit i starts up at time $k \in \{t+1,...,K(t)\}$, unit i is still up at time $t+K^i(t)$ due to min-up constraints, thus the inequality is valid.

In the (x,u,ρ) space, inequalities enforcing upper bounds on ρ^i_t are proposed in [66]

$$\rho^i_t \leq (P^i_{max} - P^i_{min})x^i_t - (P^i_{max} - SU^i)u^i_t, \quad \forall i \in \mathscr{N}\ \forall t \in \mathscr{T}$$

$$\rho^i_t \leq (P^i_{max} - P^i_{min})x^i_t - (P^i_{max} - SD^i)w^i_{t+1}, \quad \forall i \in \mathscr{N},\ \forall t \in \mathscr{T}$$

If $L^i > 1$ the following inequality holds as well

$$\rho^i_t \leq (P^i_{max} - P^i_{min})x^i_t - (P^i_{max} - SU^i)u^i_t - (P^i_{max} - SD^i)w^i_{t+1}, \quad \forall i \in \mathscr{N}, L^i > 1, \forall t \in \mathscr{T}$$

The authors of [71] introduce a polynomial class of valid inequalities strengthening ramp constraints for each unit i such that $L^i > 1$ and $RD^i > (SU^i - P^i_{min})$. In [76], the authors introduce valid inequalities for each unit i with ramp rates greater than the minimum production limit, *i.e.*, $RU^i, RD^i \geq P^i_{min}$. In [75], $\mathscr{P}^1_{ramp}$ is completely described for $T = 3$. New classes of inequalities, valid for any T, are introduced. A polynomial size extended formulation, based on the interval formulation for the MUCP, is proposed independently in [25] and in [49]. The authors of [49] show that this extended formulation can be used to generate cuts in the (x,u,p) or (x,u,ρ) space.

1.2.7 ILP formulations of the UCP

Various ILP formulations for the UCP have also been proposed in the literature (see references in surveys [1, 88]).

The authors of [3] propose an ILP model for the MUCP with ramp constraints and start-up and shut-down trajectories (as defined in Section 1.2.1). A distinction is drawn between power and energy, thus providing a more accurate description of the actual operating process of production units. The authors of [67] propose another model featuring up/down variables x as well as online/offline variables indicating that the unit is following a start-up/shut-down trajectory. This model takes into account the power provided by a unit i during its start-up and shut-down periods, even though the power output is less than P^i_{min}.

Some articles consider non-linear production cost. A linear piecewise approximation of non-convex and non-differentiable production costs, are proposed in [2]. Quadratic production costs are approximated in [13] by a piecewise linear function. Variable p_t^i is then decomposed in several variables. The authors of [26] compare the classical linear piecewise approximation for quadratic production costs to the use of perspective cuts [23]. These cuts rely on the introduction of a new variable z_t^i for each quadratic term of the form $f_t^i(x_t^i, p_t^i)$ in the objective function. The epigraph of the convex envelope of f_t^i is described by an infinite system of linear inequalities called perspective cuts.

The authors of [2] give a stairwise formulation of the start-up cost, depending on how much time a unit has been down. In [57], a model for hot and cold start-ups is presented. The authors of [48] compare various UCP formulations for one unit with s possible start-up costs, depending on the unit's down time. Start-up cost (SC) formulations featuring only up/down variables x yield a weaker relaxation value than the (x,u) SC formulation featuring both up/down and start-up variables x and u, called SC(x,u). Variables $\delta(s',i,t)$ are introduced, indicating whether unit i can incur start-up cost s' at time t. The resulting STI formulation has a better relaxation value than 3-bin. Flow formulation (F^n-*Flow*) is still integral when time-dependent start-up costs are added, as the start-up cost incurred at each time period can be directly deduced from state transitions in the network. Finally, formulation Match is introduced in [48], where flow variables are adjoined to SC(x,u) formulation. The linear relaxation obtained is better than that of formulation STI.

Many articles [13, 65–67, 71] include primary and secondary reserves in their UCP model (see survey [88]). At each time period t, the available power of unit i is $P_{max}^i x_t^i - p_t^i$. The total available power at a given time t must be greater than the primary or secondary reserve requirement R_t. To the best of our knowledge, tertiary reserve is not modeled in the literature.

CHAPTER 2

ON THE COMPLEXITY OF THE UNIT COMMITMENT PROBLEM

It is common knowledge that the UCP is hard to solve in practice. However, the only complexity result available is a reduction from the knapsack problem [86]. The latter is weakly NP-hard and most of its very large instances can be solved efficiently using commercial solvers. An interesting question is why is the UCP so hard compared to the knapsack problem. There are some trivial cases such as $\ell^i = L^i = 1$, where the UCP reduces to T independent knapsack problems. When ℓ^i and L^i are arbitrary, the min-up and min-down constraints and the demand variation over time introduce a dynamic coupling between each of these knapsack problems. If this dynamic coupling were negligible, an efficient algorithm based on the knapsack problem could be derived. The aim is to provide answers on this issue by analyzing instances capturing this dynamic coupling. We propose several results. The UCP is strongly NP-hard by reduction from the 3-partition problem, thus precluding any pseudo-polynomial dynamic programming scheme in the general case. A real-world UCP instance may not feature the characteristics of the instances used in the 3-partition reduction since such instances have distinct numerical values of power outputs and costs. We then focus on instances where the power outputs and costs are unitary. It is worth noting that in this case, the underlying knapsack instances are trivially polynomial. The MUCP is proved to be still strongly NP-hard in this context. This confirms that the dynamic coupling is by itself a major source of difficulty when solving the UCP.

Numerical experiments show that the practical difficulty to solve the MUCP increases much faster with the number of units n than with the number of time periods T. In this context, it is interesting to note that our strong NP-hardness proof of the MUCP relies on instances with n three times larger than T. Moreover, when n is fixed, we prove that a polynomial dynamic

programming scheme exists for arbitrary T. This shows that another key ingredient in the difficulty of the MUCP is the number n of dynamically coupled units. Finally, we prove that the P-IMUCP, the subproblem arising from decomposition schemes, is strongly NP-hard.

In this chapter, the MUCP is proved to be strongly NP-hard in the general case (Section 2.1). A polynomial algorithm is proposed for the IMUCP with arbitrary T whenever n is fixed (Section 2.3). The case involving a unitary cost (Section 2.4.1) and/or a unitary amount of production per unit (Section 2.4.2) remains strongly NP-hard. For each case the relative impact of parameters n and T on the complexity is discussed. Finally the P-IMUCP is shown to be strongly NP-hard (Section 2.5).

The results presented in this chapter have been published in [9].

2.1 The UCP is strongly NP-hard

In this section, we prove the following result by reduction from the 3-partition problem, proved NP-complete in [28].

Theorem 2.1. *The MUCP is strongly NP-hard for* $T = \frac{n}{3} - 1$.

Proof. Let us consider an instance of the 3-partition problem, with a set A of $3m$ integers a_1, ..., a_{3m}, a bound $B \in \mathbb{N}$ such that $\frac{B}{4} < a < \frac{B}{2}$ for all $a \in A$, and such that $\sum_{a \in A} a = mB$. The question is whether A can be partitioned into m triplets A_1, ..., A_m, such that $\sum_{a \in A_i} a = B$. Note that if a such partition of A into m subsets A_1, ..., A_m with sum B exists, then each subset A_i must contain exactly three elements.

Consider now the following instance of the MUCP. Let $T = m + 1$, with $D_t = (m - t + 1)B$, $\forall t \in \{1, ..., T\}$. Note that at each time period the demand decreases by B. Let $n = 3m$ the number of units. For each $i \in \{1, ..., 3m\}$, $P^i_{min} = P^i_{max} = a_i$; $\ell^i = T$; $L^i = 1$; $c^i_f = a_i$, $c^i_0 = c^i_p = 0$.

Let us suppose there exists a solution to the latter instance, with cost less or equal to $\sum_{i=1}^{m} iB = B\frac{m(m+1)}{2}$.

Since $\sum_{t=1}^{T} D_t = B\frac{m(m+1)}{2}$ and the unit cost is equal to the production, the cost of any solution will be at least $B\frac{m(m+1)}{2}$. If at a given time t the production is greater than the demand $D_t = (m-t+1)B$ then the solution cost will be greater than $B\frac{m(m+1)}{2}$. So for any solution of cost $B\frac{m(m+1)}{2}$, at each time period t, the units produce exactly D_t.

Let A_t be the subset of units which shut down at time t. Since every unit is up at time 1 and $\ell^i = T$, for each unit i, each unit can shut down just once so subsets A_t are disjoint. Since at each time period the units produce exactly the demand D_t, it follows $\sum_{i \in A_t} P^i_{max} = \sum_{i \in A_t} a_i = D_{t-1} - D_t = B$. Hence, the partition A_1, ..., A_T directly gives a solution to the instance of the 3-partition problem.

Conversely, from a solution to the 3-partition problem instance, a solution to this MUCP instance can be constructed with cost equal to $B\frac{m(m+1)}{2}$. ■

This result shows that the MUCP is strongly NP-hard even when each unit production cost matches its production, *i.e.*, $c_f^i = P_{min}^i = P_{max}^i$, $c_0^i = c_p^i = 0$, $\forall i \in \mathcal{N}$. As the MUCP is a particular case of the UCP, the UCP is thus strongly NP-hard as well.

2.2 The 1-period MUCP is weakly NP-hard

For $T = 1$, the MUCP is proved NP-hard from the weakly NP-hard knapsack problem. As the $T = 1$ MUCP is a knapsack problem with continuous ranges $[P_{min}^i, P_{max}^i]$, the question is whether this problem is also weakly NP-hard.

In this section we give a pseudo-polynomial algorithm for the $T = 1$ MUCP with $D_1 \in \mathbb{N}$, thus showing this problem is weakly NP-hard. We first see that given a feasible MUCP instance, there exists an optimal solution such that at most one unit i is producing neither at P_{min}^i nor at P_{max}^i. Note that the feasibility of the instance can be trivially checked.

Lemma 2.1. *For any feasible $T = 1$ MUCP instance, there exists an optimal solution $\widetilde{p}$ to the $T = 1$ MUCP, where $\widetilde{p}^i$ is the power produced by unit i, such that there exists at most one unit i such that $P_{min}^i < \widetilde{p}^i < P_{max}^i$.*

Proof. Let p be an optimal solution to the $T = 1$ MUCP, where p^i is the power produced by unit i. Suppose there are two units i and j, with $c_p^i \leq c_p^j$, such that $P_{min}^i < p^i < P_{max}^i$ and $P_{min}^j < p^j < P_{max}^j$. Let $\delta = \min(p^j - P_{min}^j, P_{max}^i - p^i)$. Consider solution $\widetilde{p}$ such that

$$\begin{cases} \widetilde{p}^i = p^i - \delta \\ \widetilde{p}^j = p^j + \delta \\ \widetilde{p}^k = p^k \qquad \text{for } k \neq i, j \end{cases}$$

By feasibility of p, solution $\widetilde{p}$ is feasible. Moreover, as $c_p^i \leq c_p^j$, solution $\widetilde{p}$ has cost less than or equal to p. As p is optimal, $\widetilde{p}$ is also optimal. Let n_p be the number of units k in solution p such that $P_{min}^k < p^k < P_{max}^k$. This number decreases to $mn_p - 1$ in solution $\widetilde{p}$. The proof is concluded with an induction argument. ■

From this property, a pseudo-polynomial dynamic programming algorithm, similar to those already existing for the knapsack problem, can be derived. For each unit i, consider the following ordering π of the units:

$$(\pi(1), \pi(2), \pi(3), ..., \pi(i), \pi(i+1), ..., \pi(n)) = (i, 1, 2, ..., i-1, i+1, ..., n)$$

Let $V^i(m, d)$ be the optimal value of the $T = 1$ MUCP instance featuring only the m first units (with respect to this ordering), with a demand d to satisfy. Moreover, for any unit $j \neq i$, the power

output p^j of j is in $\{0, P^j_{min}, P^j_{max}\}$. Then the following induction relation holds:

$$\forall m \in \{2,...,n\}, \quad V^i(m,d) = \min \begin{cases} V^i(m-1,d) \\ V^i(m-1,d-P^{\pi(m)}_{min}) + c^{\pi(m)}_f + c^{\pi(m)}_p P^{\pi(m)}_{min} \\ V^i(m-1,d-P^{\pi(m)}_{max}) + c^{\pi(m)}_f + c^{\pi(m)}_p P^{\pi(m)}_{max} \end{cases}$$

$$V^i(1,d) = \begin{cases} 0 & \text{if } d \le 0 \\ c^i_f + P^i_{min} c^i_p & \text{if } d < P^i_{min} \\ c^i_f + d c^i_p & \text{if } P^i_{min} \le d \le P^i_{max} \\ +\infty & \text{otherwise} \end{cases}$$

The optimal value v^* of the $T = 1$ MUCP is obtained as follows

$$v^* = \min_{i \in \mathcal{N}} V^i(n, D_1), \qquad \text{where } D_1 \text{ is the demand value at time 1.}$$

It follows that the $T = 1$ MUCP can be solved with an $n^2 D_1$ states dynamic programming scheme. As there also exists a reduction from the knapsack problem to the $T = 1$ MUCP, the following holds.

Theorem 2.2. *The $T = 1$ MUCP is weakly NP-hard.*

2.3 The IMUCP is polynomial when n is fixed

In the following, we consider the IMUCP where the number of units n is fixed, which is equivalent to say that n is not considered as a parameter of the problem. Note that the IMUCP is a generalization of the MUCP.

Various dynamic programming schemes exist for the UCP with only one production unit [5, 20, 24, 76]. We propose a dynamic programming algorithm for the multi-unit case, where both demand and intra-site constraints must be satisfied. The number of states is shown to be bounded by a degree n polynomial. It follows that the fixed-n IMUCP is polynomial.

For each unit $i \in \mathcal{N}$ and time period $t \in \mathcal{T}$, the possible states for unit i at time t are given by the *unit-state* set $E^i_t = \{-\ell^i, ..., -1, 1, ..., L^i\}$:

- either unit i is up at time t and must remain up for at least ϵ^i_t time periods (including t), which corresponds to the unit-state $\epsilon^i_t \in \{1, ..., L^i\}$,

- or unit i is down at t and must remain down for at least $|\epsilon^i_t|$ time periods (including t), which corresponds to the unit-state $\epsilon^i_t \in \{-\ell^i, ..., -1\}$.

Given $\epsilon^i_t \in E^i_t$, the set of the next possible unit-states for i are given by $\Gamma(\epsilon^i)$:

$$\Gamma(\epsilon^i_t) = \begin{cases} \{\epsilon^i - 1\} & \text{if } \epsilon^i_t > 1 \\ \{-\ell^i, 1\} & \text{if } \epsilon^i_t = 1 \\ \{-1, L^i\} & \text{if } \epsilon^i_t = -1 \\ \{\epsilon^i + 1\} & \text{if } \epsilon^i_t < -1 \end{cases}$$

For instance, if $\epsilon_t^i = 1$, unit i is up at time t and can at time $t+1$ either stay up or shut down for at least ℓ^i time periods.

We introduce a graph $G = (V, A)$, whose vertices are possible states of the whole n-unit system for a given time period t. An arc will be drawn between a possible state at time t and a reachable state at time $t+1$. The length of this arc will be given by the cost of the state at time $t+1$. We will show that an optimal solution to the MUCP can be obtained by finding a shortest path in this graph.

Let us define the state of the n-unit system at time t as a tuple $v_t = (\epsilon_t^1, \epsilon_t^2, ..., \epsilon_t^n)$ where $\epsilon_t^i \in E_t^i$. If demand D_t is met when all units which are up in v_t produce at P_{max}, then tuple v_t is said to fulfill the demand. Moreover, if for each site Σ, there is at most one unit i in Σ such that $\epsilon_t^i = L^i$, then v_t fulfills intra-site constraints. Let V_t be the set of all tuples v_t which correspond to possible states, *i.e.*, states that both fulfill the demand and intra-site constraints. We then set $V = \cup_{t=1}^{T} V_t \cup \{v_0, v_{T+1}\}$ where v_0 is a source vertex and v_{T+1} is a sink vertex.

For any $t \in \{0, ..., T-1\}$, there is an arc between a state $v_t = (\epsilon_t^1, \epsilon_t^2, ..., \epsilon_t^n) \in V_t$ and $v_{t+1} = (\epsilon_{t+1}^1, \epsilon_{t+1}^2, ..., \epsilon_{t+1}^n) \in V_{t+1}$ if and only if for all $i \in \mathcal{N}$, $\epsilon_{t+1}^i \in \Gamma(\epsilon_t^i)$. Moreover, A contains an arc (v_T, v_{T+1}) for any $v_T \in V_T$.

The length of an arc is given by $\lambda : A \to \mathbb{R}_+$. For each $\epsilon \in V$, $\lambda(\epsilon, v_{T+1}) = 0$. For each arc (v_t, v_{t+1}), $\{0, ..., T-1\}$, the length is given by:

$$\lambda(v_t, v_{t+1}) = \sum_{i \text{ up in } v_{t+1}} c_f^i + \sum_{i \text{ starts up in } v_{t+1}} c_0^i + \sum_{i \text{ up in } v_{t+1}} c_p^i \overline{p}_{t+1}^i$$

where for each t, $(\overline{p}_t^i)_{i \text{ up in } v_t}$ is the optimal solution to the following production dispatch LP:

$$\begin{aligned} \min_{p_t^i \in \mathbb{R}} \quad & \sum_{i \in X_t} c_p^i p_t^i \\ \text{s. t.} \quad & \sum_{i \in X_t} p_t^i \geq D_t \\ & P_{min}^i \leq p_t^i \leq P_{max}^i \quad \forall i \text{ up in } v_t \end{aligned}$$

This problem can be solved in linear time, provided that the units are sorted in non-decreasing order of c_p^i. Indeed, the optimal solution can be constructed by generating the maximum quantity of power possible with the lowest cost units, until the demand is met. Note that, by the construction of v_t, this production dispatch LP is always feasible.

Considering graph $G = (V, A)$ and length λ, a solution to the fixed-n MUCP is exactly a solution to the shortest path problem from v_0 to v_{T+1} in graph G. This leads to the following theorem.

Theorem 2.3. *The fixed-n IMUCP can be polynomially solved in* $O(T^2 \prod_{i=1}^{n} (L^i + \ell^i)^2)$.

Proof. For each unit i and t, $|E_t^i| \leq L^i + \ell^i$. Thus, $|V_t| \leq \Pi_{i=1}^n (L^i + \ell^i)$. It follows that the number of vertices in G is bounded by $2 + T\Pi_{i=1}^n (L^i + \ell^i)$. As n is fixed, this corresponds to a polynomial number of vertices. Since graph G is acyclic, a shortest path can be computed using Bellman algorithm. ■

Note that Theorem 2.3 is proved for units with no ramp constraints. Depending on the power outputs nature, the result is twofold. If for each unit i, the feasible power outputs are in a compact bounded set $\mathscr{P}^i$, this dynamic programming scheme cannot be extended to take ramp constraints into account. Other approaches must be considered in order to find out whether the corresponding fixed-n problem is polynomial. If for each unit i, the feasible power outputs are in a finite set $\mathscr{P}^i$, the proof of Theorem 2.3 can be extended to show that this problem is also polynomial when n is fixed. The UCP solved daily at EDF [18, 80] corresponds to the latter case, which features the following constraints:

$$(RF) \begin{cases} \text{Ramp constraints} \\ \text{Finite set } \mathscr{P}^i \text{ of feasible power outputs, for each unit } i. \end{cases}$$

Theorem 2.4. *The fixed-n IMUCP with constraints* (RF) *can be polynomially solved in* $O(T^2 \prod_{i=1}^{n} (L^i |\mathscr{P}^i| + \ell^i)^2)$.

Indeed, the set of possible states for a unit at time t is then a subset of $E_t^i \times \mathscr{P}^i$. The ramp constraints are taken into account by allowing arcs between two states s_1 and s_2 only if the ramp constraints are satisfied when the n-unit system switches from s_1 to s_2. In this case, the number of vertices in graph G is bounded by $2 + T\Pi_{i=1}^n (L^i |\mathscr{P}^i| + \ell^i)$. It follows that the problem can be solved in $O(T^2 \prod_{i=1}^{n} (L^i |\mathscr{P}^i| + \ell^i)^2)$ time. Note that this bound is not reached in practice since many states are not attainable due to the ramp constraints.

2.4 NP-hard special cases of the MUCP

When $T = 1$, the MUCP is weakly NP-hard by reduction from the knapsack problem [86]. However the knapsack problem becomes polynomial if either the item cost or the item weight is unitary. Interestingly, it is shown in this section the corresponding MUCP cases are NP-hard. This highlights that the difficulty of the MUCP does not only lie in the knapsack reduction at $T = 1$, but also in the combinatorial aspects introduced by the min-up/min-down time constraints for more general time horizons. Contrary to the general case of Theorem 2.1, we have only proved that these easier cases are strongly NP-hard for T much greater than n.

2.4.1 The unit-cost MUCP is NP-hard

We define the *unit-cost* MUCP, as a particular case of the MUCP where $c_0^i = c_p^i = 0$ and $c_f^i = 1$, for each unit i. This problem can be solved in polynomial time when $T = 1$, by sorting the units in

decreasing order w.r.t. their P^i_{max}.

Theorem 2.5. *The unit-cost MUCP is NP-hard for $T = n+1$ and strongly NP-hard for $T = \frac{1}{3}n^2$.*

Proof. Let us consider an instance of the partition problem, with a set A of n positive integers $a_1, ..., a_n$. The question is whether A can be partitioned into two subsets A_1 and A_2 such that $\sum_{i\in A_1} a_i = \sum_{i\in A_2} a_i$. Note that if such a partition exists, then $\sum_{i\in A_1} a_i = \sum_{i\in A_2} a_i = B$ where $B = \frac{1}{2}\sum_{i\in A} a_i$.

Consider now the following instance of the unit-cost MUCP: let $T = n+1$ with $D_1 = D_T = B$ and $D_t = 0$, for all $t \in [2, T-1]$. Let us define n units such that $P^i_{max} = P^i_{min} = a_i$, $\ell^i = T$, $L^i = 1$, $i \in \{1, ..., n\}$. Assume there exists a solution to the latter instance with cost less than or equal to n. In such a solution, each unit up at time 1 must be down at time T. Indeed, $\ell^i = T$ so when a unit shuts down it can never start up again. However, if a unit i remains up from time 1 to time T, then the cost of solution S is at least $n+1$, which is a contradiction. Thus, let A_1 be the set of units up at time 1, and A_2 be the set of units up at time T. The claim is that (A_1, A_2) gives a solution to the instance of the partition problem. Indeed, A_1 and A_2 are disjoint, as all units up at time 1 are down at time T. Moreover, the units in A_1 satisfy the demand at time 1, so $\sum_{i\in A_1} a_i \geq B$. Similarly, $\sum_{i\in A_2} a_i \geq B$. As A_1 and A_2 are disjoint, $2B \leq \sum_{i\in A_1} a_i + \sum_{i\in A_2} a_i \leq \sum_{i\in A} a_i = 2B$, we get $\sum_{i\in A_1} a_i = B$, $\sum_{i\in A_2} a_i = B$ and $A_1 \cup A_2 = A$.

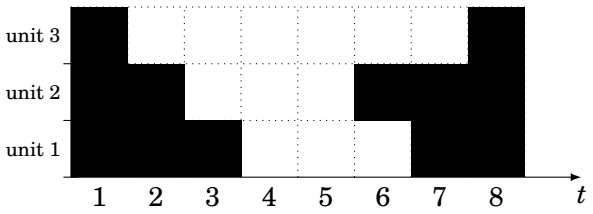
Conversely, any solution to the instance of the partition problem can similarly be used to construct a solution to this unit-cost MUCP instance with cost n.

This transformation can be slightly modified to show that the unit-cost MUCP is strongly NP-hard, by reduction from the 3-partition problem. Let us consider a 3-partition problem instance with a set A of $3m$ integers a_1, ..., a_{3m}, a bound $B \in \mathbb{N}$ such that $B/4 < a < B/2$ for all $a \in A$, and such that $\sum_{a\in A} a = mB$. We consider an instance of the unit-cost MUCP with $n = 3m$ units, time horizon $T = mn+1$ and demand $D_{kn+1} = B$ for each $k \in \{0, ..., m\}$ and $D_t = 0$ otherwise. The units have the same characteristics as those of the reduction from the partition problem. We can similarly prove that there is a solution to the 3-partition problem if and only if there is a solution to this instance of the unit-cost MUCP. ■

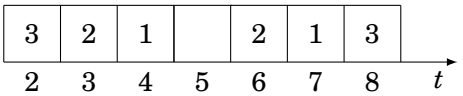
2.4.2 The unit-power MUCP is strongly NP-hard

We define the *unit-power* MUCP as a particular case of the MUCP where all units which are up produce the same amount of power P such that $P^i_{min} = P^i_{max} = P$, $i \in \mathcal{N}$. The unit-power MUCP can be solved in polynomial time when $T = 1$, by sorting the units in increasing order w.r.t. their costs.

The unit-power MUCP is shown to be strongly NP-hard for arbitrary T, by reduction from the single machine Flow-Shop Problem with minimum delays and unit-time operations (FSP). This problem, proved strongly NP-hard in [92], is defined as follows.



(a) Solution to the unit-power MUCP instance corresponding to I_{ex}



(b) Solution to the FSP instance I_{ex}

Figure 2.1: Instance I_{ex} of the FSP: $|J| = 3$, $p = [3,3,6]$ and $\Delta = 7$

INSTANCE: Consider a set of jobs J, such that each job consists of two unit-time operations. The two operations of a given job j must be scheduled with an intermediate delay p_j. Let Δ be an integer.

QUESTION: A schedule (σ_1, σ_2) is defined by function $\sigma_1 : J \to \mathbb{Z}_0^+$ (resp. $\sigma_2 : J \to \mathbb{Z}_0^+$) that gives the schedule of operation 1 (resp. 2) for each job. For all $t \geq 1$, there is at most one job $j \in J$ such that $\exists i \in \{1,2\}$, $\sigma_i(j) = t$. Moreover, $\sigma_1(j) + p_j \leq \sigma_2(j)$ for all $j \in J$. Is there a schedule (σ_1, σ_2) with a makespan less than or equal to Δ, *i.e.* $\sigma_2(j) \leq \Delta$, for all $j \in J$?

Figure 2.1b shows a solution to the following FSP instance, denoted by I_{ex}: $|J| = 3$, $p = [3,3,6]$ and $\Delta = 7$. The first operation of job 3 is executed at time 1, and the second operation of job 3 at time 7. The intermediate delay $p_3 = 6$ is thus respected. Similarly, first and second operations of jobs 1 and 2 are executed before the deadline while satisfying their intermediate delays.

We first give a technical lemma discussing properties of an FSP solution.

Lemma 2.2. *From any solution to a given FSP instance, another solution can be constructed such that all first operations are executed at times* $\{1,...,|J|\}$*, and all second operations are executed at times* $\{\Delta - |J| + 1, ..., \Delta\}$.

Proof. If there was a first operation executed after a second operation, then the two operations could be permuted without reducing the delay between the first and second operations of any job. Now suppose the first (resp. second) operation of a given job j is executed after time $|J|$ (resp. before time $\Delta - |J| + 1$). Since all first operations precede all second operations then there is an idle time period $t \in \{1,...,|J|\}$ (resp. $\{\Delta - |J| + 1, ..., \Delta\}$) at which no operation is executed. Thus the execution of the first (resp. second) operation of j can be scheduled at time t without increasing the delay between the first and the second operation of job j. ■

We now prove the unit-power MUCP is strongly NP-hard by reduction from the FSP. This reduction holds in the case where both power and cost are unitary. This proves that the corresponding problem, denoted by unit-(power+cost) MUCP, is strongly NP-hard. Contrary to Theorem 2.5 for the unit-cost MUCP, this result does not provide a real measure of the respective role of parameters n and T toward the problem's complexity.

Theorem 2.6. *The unit-power MUCP is strongly NP-hard.*

Proof. Let consider an instance I_{FSP} of the FSP problem. We construct an instance I of the unit-power MUCP as follows: let $n = |J|$ units and a time horizon $T = \Delta + 1$. For each unit $j \in J$, let $L^j = T$ and $\ell^j = p^j$; $c_0^j = c_p^j = 0$ and $c_f^j = 1$. Note that since there is a single machine, and two unit-time operations per job, if $\Delta < 2n$ then there is no solution to I_{FSP}. We thus suppose $\Delta \geq 2n$. The demand D_t is given by:

$$D_t = \begin{cases} n - t + 1 & \text{if } t \in \{1, ..., n\} \\ 0 & \text{if } t \in \{n+1, ..., T-n\} \\ t - (T-n) & \text{if } t \in \{T-n+1, ..., T\} \end{cases}$$

The claim is that if we can find a solution S of cost at most $n(n+1)$ for instance I then we can find a schedule for instance I_{FSP}. Let us consider a solution S with cost at most $n(n+1)$. First note that $\sum_{t=1}^{T} D_t = n(n+1)$ and then any solution to I is with cost at least $n(n+1)$, since each up unit costs 1 per time period.

In solution S, exactly D_t units are then up at time t, for each $t \in \{1, ..., T\}$. As $L^j = T$ for each $j \in \{1, ..., n\}$, each unit shuts down at most once on the time horizon T. As $D_1 = n$ and $D_{n+1} = 0$, each unit must shut down exactly once on the time horizon. Similarly, as the demand at time T is n, each unit starts up exactly once. For a unit j, let $\sigma_1(j)$ (resp. $\sigma_2(j)$) be the shut-down (resp. start-up) time period of unit j in solution S.

For a solution S to instance I, we construct a solution S_{FSP} to instance I_{FSP} in which the first operation of job j is processed at time $\sigma_1(j)$ and the second operation of job j is processed at time $\sigma_2(j)$.

Figure 2.1 depicts a reduction from the FSP instance I_{ex} to a unit-power MUCP instance. Figure 2.1b shows the solution to the FSP instance I_{ex}, while Figure 2.1a shows the solution to the corresponding MUCP instance.

Since unit j has minimum down-time p_j, for all $j \in J$, $\sigma_1(j) + p_j \leq \sigma_2(j)$ holds. Moreover, as $D_T = n$, $\sigma_2(j) \leq \Delta$ holds. The claim is there is at most one shut-down (resp. start-up) per time period. Indeed, since $D_{n+1} = 0$ (resp. $D_{T-n} = 0$) and each unit shuts down (resp. starts up) exactly once, all the shut-downs (resp. start-ups) happen between times 2 and $n+1$ (resp. $T-n+1$ and T). If there were two units shutting-down (resp. starting-up) at the same time period $t \in \{2, ..., n+1\}$ (resp. $\{T-n+1, ..., T\}$), then, as $D_t = D_{t-1} - 1$ (resp. $D_t = D_{t-1} + 1$), either the demand would not be satisfied at time t (resp. $t-1$), or there would be more than D_{t-1} (resp. D_t) units up at time $t-1$ (resp. t), which would be a contradiction. Consequently, at most one operation is executed per time period.

Conversely, if there is a solution S_{FSP} to instance I_{FSP}, a solution S to instance I of cost at most $n(n+1)$ can be constructed. Let S_{FSP} be a solution of the FSP instance. From Lemma 2.2, we can suppose that in S_{FSP} all first operations are executed at times $\{2, ..., n+1\}$, and all second operations are executed at times $\{T-n+1, ..., T\}$. From this solution S_{FSP}, we compute a solution of MUCP instance S, by shutting-down (resp. starting-up) unit j at time $\sigma_1(j)$ (resp. $\sigma_2(j)$).

This shows that the FSP problem can be polynomially transformed to the unit-power MUCP.
■

2.5 The P-IMUCP is strongly NP-hard

In this section, the P-IMUCP is considered. Recall the demand is no longer to be satisfied, while the fixed production cost $\pi^{i,t}$ and the proportional production cost $\rho^{i,t}$ can be negative and depend on the time period. For a given value $K \in \mathbb{R}$, the P-IMUCP is to decide whether there is a plan satisfying minimum up and down time constraints, with cost at most K and such that there is at most one start-up per time t. We show this problem is strongly NP-hard by reduction from a restricted version of the Satisfiability problem, denoted by R3-SAT. Problem R3-SAT is such that there are at most three variables per clause, and each variable is restricted to appear once negatively and once or twice positively overall in the set of clauses C. This problem has been proved to be strongly NP-hard [77].

Theorem 2.7. *The P-IMUCP is strongly NP-hard.*

Proof. Consider an instance of R3-SAT with clauses c_1, ..., c_q and variables x_1, ..., x_p. Consider the following instance of the P-IMUCP, with time horizon $T = 6p$ and n units where $n = p + q$. Each unit $i \in \{1,...,p\}$ is associated to variable x_i while each unit $p+k$, $k \in \{1,...,q\}$, is associated to clause c_k. For each unit $i \in \{1,...,p+q\}$, $L^i = 1$ and $c_0^i = \rho^{i,t} = 0$.
For each unit $i \in \{1,...,p\}$, $\ell^i = 2p$ and

$$\pi^{i,t} = \begin{cases} -\frac{1}{2} & \text{if } t = 2i-1 \text{ or } t = 4p+2i-1 \\ -1 & \text{if } t = 2p+2i-1 \\ 2 & \text{otherwise.} \end{cases}$$

For each unit $p+k$, $k \in \{1,...,q\}$, $\ell^{p+k} = T$. Recall that a given variable x_s, $s \in \{1,...,p\}$ appears once or twice (resp. once) positively (resp. negatively) from c_1 to c_q. To compute $\pi^{p+k,t}$, we construct an auxiliary time value $\mu_q(X)$ associated to every literal $X \in c_k$:

$$\mu_q(X) = \begin{cases} 2s-1 & \text{if } X = x_s \text{ appearing positively} \\ & \text{for the first time} \\ 4p+2s-1 & \text{if } X = x_s \text{ appearing positively} \\ & \text{for the second time} \\ 2p+2s-1 & \text{if } X = \overline{x}_s. \end{cases}$$

$$\pi^{p+k,t} = \begin{cases} -1 & \text{if } \exists\, X \in c_k \text{ such that } \mu_q(X) = t \\ 3 & \text{otherwise.} \end{cases}$$

To illustrate, consider an R3-SAT instance I_R with $p = 3$, $q = 2$, $c_1 = \overline{x}_1 \vee x_2 \vee x_3$, $c_2 = x_1 \vee x_2 \vee \overline{x}_3$. Figure 2.2 shows costs $\pi^{i,t}$ in the corresponding instance of the P-IMUCP.

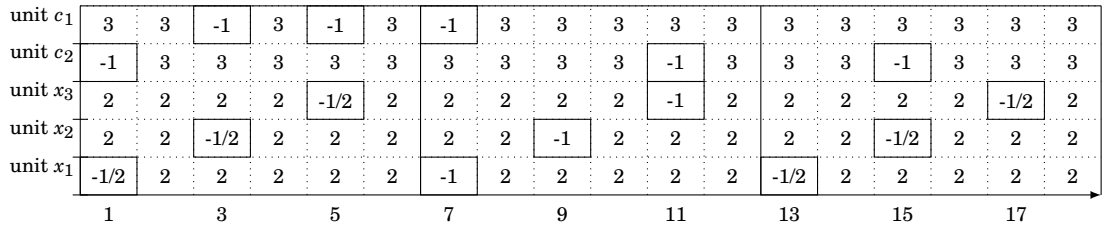


Figure 2.2: Cost $\pi^{i,t}$ for the P-IMUCP instance corresponding to R3-SAT instance I_R

We will prove that there is a solution to R3-SAT if and only if there is a solution with cost at most $-n$ to this instance of the P-IMUCP. First, consider solution S with cost at most $-n$ to this P-IMUCP. The claim is the total cost of a unit $i \in \{1,...,p\}$ is at least -1. Indeed, each unit $i \in \{1,...,p\}$ has a negative cost at times $2i-1$, $2p+2i-1$ and $4p+2i-1$, and a positive cost of 2 at any other time. If unit i were up only at times where its cost is negative, unit i would contribute $-1/2-1-1/2=-2$ to the solution cost. If unit i is up at any time where the cost is positive, its contribution to the solution cost is at least 0. Because of the minimum down time $\ell^i = 2p$, unit i cannot start up at time $2i-1$, shut down at time $2i$, and start up again at $2p+2i-1$. Therefore unit i contributes at least -1 to the solution cost, and contributes exactly -1 in one of the following two cases:
(i) Unit i is up at times $2i-1$ and $4p+2i-1$, and down at all other times
(ii) Unit i is up at time $2p+2i-1$ and down at all other times.

Similarly, the claim is the total cost of a unit $p+k$, $k \in \{1,...,q\}$, is at least -1. Indeed, each unit $p+k$, $k \in \{1,...,q\}$, has cost -1 at times $\mu_k(X)$, for each literal X in c_k. Since there are at most three variables per clause, there are at most three time periods where the cost of unit $p+k$ is -1. Since at all other times, the cost of unit $p+k$ is 3, if unit $p+k$ is up at one given time where its cost is positive, its contribution to the cost would be 0 at least. Moreover, unit $p+k$ can start up just once on the time horizon because of the minimum down time $\ell^{p+k} = T$. Since all times $\mu_k(X)$, $X \in c_k$, are odd, if unit $p+k$ is up only at times $\mu_k(X)$ it has to be up at one given time $\mu_k(X)$ and down at all other times. In this case its cost contribution is -1.

Since solution S has cost at most $-n$, each unit contributes exactly -1 to the solution total cost. We construct a solution to the R3-SAT instance such that variable x_i has truth value “false" in case (i), and truth value “true" in case (ii). Each unit $p+k \in \{p+1,...,p+q\}$ starts up once, meaning there is at least one true literal X in clause c_k. Otherwise the unit associated to X would have started up at the same time, contradicting the intra-site constraints.

Conversely, any solution to R3-SAT can be transformed to a solution to this P-IMUCP with cost $-n$. ■

<table>
<tr><th></th><th>Polynomial</th><th>NP-hard</th><th>Strongly NP-hard</th></tr>
<tr><td>MUCP with $T = 1$</td><td>(fixed n, $T = 1$)</td><td rowspan="2">(n, $T = 1$)</td><td>$\varnothing$</td></tr>
<tr><td>MUCP
IMUCP
IMUCP with (RF)</td><td>(fixed n, T)</td><td>(n, $T = \frac{1}{3}n$)</td></tr>
<tr><td>unit-cost MUCP</td><td>(n, $T = 1$)</td><td>(n, $T = n + 1$)</td><td>(n, $T = \frac{1}{3}n^2 + 1$)</td></tr>
<tr><td>unit-(power+cost) MUCP</td><td rowspan="2"></td><td colspan="2">($n = |J|$, $T = \Delta$)$^{\circ}$</td></tr>
<tr><td>P-IMUCP</td><td>(fixed n, T)</td><td colspan="2">($n = p + k$, $T = 6p$)$^{\diamondsuit}$</td></tr>
</table>

Summary of the complexity results

Table 2.1: $\circ$: where $|J|$ is the number of jobs and Δ the deadline of the FSP
$\diamondsuit$: where p is the number of variables and k the number of clauses of R3-SAT.

Conclusion

The UCP is NP-hard in the strong sense. This explains better than the classical knapsack reduction the computational challenge to solve the UCP in practice. In particular, it shows that the knapsack aspects in the UCP are not the only source of difficulty. On the contrary, this result confirms a common finding that the so-called dynamic coupling has a large impact on the UCP complexity. By studying special cases of the UCP which do not feature any knapsack-related difficulty, we proved that the coupling of unitary demands with minimum up and down time constraints represents one major source of difficulty. This implies that the combinatorial aspects introduced by these constraints should be specifically tackled when solving the UCP. Interestingly, the IMUCP (with or without constraints (RF)) can be solved in polynomial time whenever n is fixed, regardless of parameters T, L, ℓ. This highlights the major impact of parameter n, compared to other parameters, with respect to the problem's complexity. Finally we have shown that the P-IMUCP – the subproblem arising in practice when the UCP is solved through a decomposition method – is strongly NP-hard for a subset of units.

The complexity results are summarized in Table 2.1. Each entry row-wise is associated to one of the problem studied. In the first entry column-wise, some polynomial cases of the problem are listed, the second (resp. third) entry column-wise lists cases in which the problem is NP-hard in the ordinary (resp. strong) sense.

Note that the unit-cost MUCP becomes NP-hard as soon as $T = n+1$. It even becomes strongly NP-hard as soon as $T = \frac{1}{3}n^2 + 1$, meaning that one cannot expect to find a polynomial time algorithm that could be applied in practice when the units considered have similar costs. In particular, the result holds even when each unit has a single feasible power output level (implying that its cost depends only on its up/down status). The same remark goes for the unit-power MUCP: even if the units considered have similar power outputs, solving the UCP will anyway be computationally challenging.

Given these complexity results, some perspectives for future work would be to determine which instances will be hard to solve in practice. For example, it would be useful to determine

for which instances the dynamic programming scheme provided for the fixed-n IMUCP remains tractable. With an increasing competition on electricity markets, another perspective would be to consider the complexity of the P-IMUCP when revenues come from trading the site's production. The problem (also referred to as self-UCP) is then to find a production plan for each unit, and a subset of so-called power products to trade with delivery patterns. It would be interesting to analyze the complexity of this problem even in the 1-unit case, as the difficulty will come from the structure of the power products considered.

Part I

Polyhedral combinatorics of the MUCP

CHAPTER 3

THE MIN-UP/MIN-DOWN UNIT COMMITMENT POLYTOPE

From the complexity results in Chapter 2, it appears that the difficulty of the MUCP arises in particular from the dynamic coupling of a large number n of units. Indeed, the MUCP is polynomial when n is fixed and T arbitrary, but is already NP-hard when $T = 1$ and n is arbitrary.

Several polyhedral studies for the 1-unit UCP can be found in the literature (see Section 1.2.6). When n-unit are considered, the demand constraints intersect the 1-unit polytopes with T knapsack polytopes, thus making the polyhedral structure more intricate. To the best of our knowledge, no polyhedral study for the n-unit case is proposed in the literature.

In this chapter, we investigate some polyhedral aspects of the MUCP with n production units. As a preliminary, we show in Section 3.1 that all demand-coupling formulations for the n-unit MUCP, *i.e.*, formulations such that the only coupling inequalities are demand constraints, have same linear relaxation value. We thus choose to focus our study on the polytope defined by formulation $F^n_{x,u}$, which features the most natural decision variables of the problem. In Section 3.2, some facial properties of inequalities (1.2), (1.3) and (1.4) are given. In Section 3.3, the rank of a subset $C \subset \mathcal{N}$ at time $t \in \mathcal{T}$ is defined as the minimum number of units of C up at time t. This rank accounts for the satisfaction of the demand constraint at time t alongside with min-up and min-down time constraints. On this basis, we describe in Section 3.4 a large family of valid inequalities generalizing both min-up/min down inequalities and extended cover inequalities from the knapsack polytope. Among them, up-set inequalities are directly translated from the knapsack polytope. Interval up-set inequalities, relying on the rank of a given C, are more dedicated to the UCP as they capture the coupling of demand and min-up/min-down time constraints. Facet defining cases are studied in Section 3.5.

The results presented from Section 3.2 to 3.5 have been published in [8].

3.1 Comparison of demand-coupling formulations

In the 1-unit case, formulations $(F^1_{x,u})$, $(F^1\text{-}Flow)$ and $(F^1\text{-}Int)$ define integral polytopes. The integrality of formulation $(F^1_{x,u})$ (see Theorem 1.5) has been proven in [79]. In Section 3.1.2, we give another proof of this result by showing that the linear relaxation value of $F^1_{x,u}$ is equal to the linear relaxation value of $(F^1\text{-}Flow)$. As $(F^1\text{-}Flow)$ is integral in the 1-unit case, the result will follow directly from Theorem 1.1.

In the n-unit case, the integrality property is lost, due to demand the constraints coupling the units. Recall from Section 1.2.5 that a demand-coupling formulation is an MUCP formulation such that the only coupling inequalities are demand constraints:

$$
\begin{array}{lll}
(F^n_{dc}) \quad \min\limits_{z,p} & \boldsymbol{c_z z} + \boldsymbol{c_p p} & \\
\text{s. t.} & (z^i, p^i) \in \Pi^i_F & \forall i \in \mathcal{N} \\
& \sum\limits_{i=1}^{n} p^i_t \geq D_t & \forall t \in \mathcal{T} \\
& z^i \in \mathbb{Z}^m,\ p^i \in \mathbb{R}^p & \forall i \in \mathcal{N}
\end{array}
$$

where (c_z, c_p) is the cost vector and $\Pi^i_F = \{A^i z^i + B^i p^i \leq d^i\}$ is a polyhedron such that $\Pi^i_F \cap (\mathbb{Z}^m \times \mathbb{R}^p)$ is a set of feasible plans for unit i, expressed with arbitrary variables $z^i \in \mathbb{Z}^m$ and production variables p^i_t. In particular, flow and interval formulations $(F^n\text{-}Flow)$ and $(F^n\text{-}Int)$ are demand-coupling formulations.

In the following, for a given formulation (F), the linear relaxation value of (F) is denoted by $v(F)$. We show in Section 3.1.1 that any demand-coupling formulation (F^n_{dc}) for the n-unit MUCP has no better linear relaxation than formulation $(F^n_{x,u})$, *i.e.*, $v(F^n_{x,u}) \geq v(F^n_{dc})$. In the case of flow and interval formulations $(F^n\text{-}Flow)$ and $(F^n\text{-}Int)$, the relaxation values are equal, *i.e.*, $v(F^n_{x,u}) = v(F^n\text{-}Flow) = v(F^n\text{-}Int)$.

Note that in this section, for any $n \in \mathbb{N}$, the cost vector of formulation $(F^n_{x,u})$ is completely arbitrary, *i.e.* cost coefficient of x^i_t (resp. u^i_t) is $c^i_t \in \mathbb{R}$ (resp. $c^i_{0,t} \in \mathbb{R}$), $i \in \mathcal{N}$, $t \in \mathcal{T}$. The cost vectors of formulations $(F^n\text{-}Flow)$ and $(F^1\text{-}Int)$ are modified accordingly.

3.1.1 Comparison of demand-coupling formulations for the n-unit MUCP

Now consider a demand-coupling formulation (F^n_{dc}) for the n-unit MUCP.

We will prove that the relaxation value of (F^n_{dc}) is always less than or equal to the relaxation value of $(F^n_{x,u})$.

First consider the 1-unit min-up/min-down and production limit polytope

$$
P^1_{x,u,p} = \left\{ (x^1, u^1, p^1) \mid x^1_t, u^1_t \in [0,1],\ \text{s.t. (1.2), (1.3), (1.4), (1.6)} \right\}.
$$

Lemma 3.1. *Any extreme point* $(\overline{x},\overline{u},\overline{p})$ *of* $P^1_{x,u,p}$ *is such that* $\overline{x}\in\mathbb{Z}^T$, $\overline{u}\in\mathbb{Z}^{T-1}$.

Proof. If $\overline{x}$ and $\overline{u}$ had fractional components, as $(\overline{x},\overline{u})\in P^1_{x,u}$, vector $(\overline{x},\overline{u})$ could be written as a combination of extreme points $(x(1),u(1))$, ..., $(x(k),u(k))$ of $P^1_{x,u}$, *i.e.*,

$$(\overline{x},\overline{u})=\sum_{k'=1}^{k}\lambda_{k'}(x(k'),u(k')),\qquad \text{where } \sum_{k'=1}^{k}\lambda_{k'}=1 \text{ and } \lambda_{k'}\neq 1,\ \forall k'\in\{1,...,k\}$$

By Theorem 1.5, $(x(1),u(1))$, ..., $(x(k),u(k))$ are integer points. For $k'\in\{1,...,k\}$, consider point $(x(k'),u(k'),p(k'))\in P^1_{x,u,p}$ where for each $t\in\mathcal{T}$

$$p(k')_t=\begin{cases}\frac{\overline{p}_t}{\overline{x}_t} & \text{if } x(k')_t=1\\ 0 & \text{otherwise}\end{cases}$$

Note that due to production limits, $\frac{\overline{p}_t}{\overline{x}_t}\in[P^1_{min},P^1_{max}]$. Then

$$(\overline{x},\overline{u},\overline{p})=\sum_{k'=1}^{k}\lambda_k(x(k'),u(k'),p(k'))$$

As $\sum_{q=1}^{k}\lambda^k=1$ and $\lambda'_k\neq 1$, $\forall k'\in\{1,...,k\}$, $(\overline{x},\overline{u},\overline{p})$ is not an extreme point of $P^1_{x,u,p}$. ■

Now we prove the following result.

Theorem 3.1. $v(F^n_{x,u})\geq v(F^n_{dc})$

Proof. Consider a solution $(\overline{x},\overline{u},\overline{p})$ of $F^n_{x,u}$. For each $i\in\mathcal{N}$, $(\overline{x}^i,\overline{u}^i,\overline{p}^i)$ belongs to polytope $P^1_{x,u,p}$ defined with characteristics (L^i, ℓ^i, P^i_{min}, P^i_{max}) from unit i. Denoting by P the extreme point set of $P^i_{x,u,p}$,

$$(\overline{x}^i,\overline{u}^i,\overline{p}^i)=\sum_{q\in P}\lambda_q q$$

where $\sum_{q\in P}\lambda^q=1$. By Lemma 3.1, points $q\in P$ have integer x and u components, and thus represent feasible plans for unit i. Therefore, for each $q\in P$, there exists a solution $\phi(q)\in\Pi^i_F\cap(\mathbb{Z}^m\times\mathbb{R})$ representing the same feasible plan for unit i. In particular, the solution cost is the same, as well as the quantity of power produced by unit i at each time period t.

We then define $\phi(\overline{x}^i,\overline{u}^i,\overline{p}^i)$

$$\phi(\overline{x}^i,\overline{u}^i,\overline{p}^i)=\sum_{q\in P}\lambda_q\phi(q)$$

As Π^i_F is convex, $\phi(\overline{x}^i,\overline{u}^i,\overline{p}^i)\in\Pi^i_F$. Moreover, since $\sum_{q\in P}\lambda^q=1$, the cost of $\phi(\overline{x}^i,\overline{u}^i,\overline{p}^i)$ is equal to the cost of $(\overline{x}^i,\overline{u}^i,\overline{p}^i)$, and the quantity of power produced by unit i at each time period is the same in both solutions. Therefore, $(\phi(\overline{x}^i,\overline{u}^i,\overline{p}^i))_{i\in\mathcal{N}}$ is a solution to the linear relaxation of (F^n_{dc}) with same cost as $(\overline{x},\overline{u},\overline{p})$. This concludes the proof. ■

This shows that no MUCP formulation of the form (F^n_{dc}) can have a better linear relaxation value that $(F^n_{x,u})$. In particular, this shows that neither formulation $(F^n\text{-}Int)$ nor $(F^n\text{-}Flow)$ improves the relaxation value of $(F^n_{x,u})$.

Note that for any solution (f,g,p) of the linear relaxation of $(F^n\text{-}Flow)$, a solution (x,u,p) of $(F^n_{x,u})$ with same cost can be defined as follows. For each $i \in \mathcal{N}$, $t \in \mathcal{T}$

$$x^i_t = \sum_{t'=1}^{t} \sum_{t''=t+1}^{T+1} f^i(t',t'')$$
$$u^i_t = \sum_{t''=t+1}^{T+1} f^i(t,t'')$$

Similarly, for any solution (y,p) of the linear relaxation of $(F^n\text{-}int)$, a solution (x,u,p) of $(F^n_{x,u})$ with same cost be constructed as follows. For each $i \in \mathcal{N}$, $t \in \mathcal{T}$

$$x^i_t = \sum_{\{t_0,\dots,t_1-1\} \in \mathcal{Y}_i \mid t \in \{t_0,\dots,t_1-1\}} y^i(t_0,t_1)$$
$$u^i_t = \sum_{\{t,\dots,t_1-1\} \in \mathcal{Y}_i} y^i(t_0,t_1)$$

Corollary 3.1.

$$v(F^n_{x,u}) = v(F^n\text{-}Flow) = v(F^n\text{-}Int)$$

All demand-coupling formulations proposed in Section 1.2.5 (*i.e.*, formulations $(F^n_{x,u})$, $(F^n\text{-}Flow)$ and $(F^n\text{-}Int)$) are equivalent from a linear relaxation point of view. However, formulations $(F^n\text{-}Flow)$ and $(F^n\text{-}int)$ may be harder to manage as they feature $\theta(nT^2)$ binary variables, while $(F^n_{x,u})$ features $2nT$ binary variables. In the rest of the chapter, we will thus focus our polyhedral study on the polytope defined by formulation $(F^n_{x,u})$.

3.1.2 Integrality of $(F^1\text{-}Flow)$

In this section, we prove that $v(F^1_{x,u}) = v(F^1\text{-}Flow)$. First, it can be readily checked that

Theorem 3.2. $v(F^1_{x,u}) \leq v(F^1\text{-}Flow)$

Proof. Let $(\overline{f}^1, \overline{g}^1)$ be a solution of $F^1\text{-}Flow$. Then consider solution $(\overline{x}^1, \overline{u}^1)$ such that

$$\begin{array}{lcll} x^1_t & = & \sum_{t'=1}^{t} \sum_{t''=t+1}^{T+1} f^1(t',t'') & \forall t \in \mathcal{T} \\ u^1_t & = & \sum_{t''=t+1}^{T+1} f^1(t,t'') & \forall t \geq 2 \end{array}$$

As $\overline{f}^1$ and $\overline{g}^1$ are nonnegative, $\overline{x}^1$ and $\overline{u}^1$ are nonnegative too. Let $t \leq L^1 + 1$. We check that the min-up constraint at t is satisfied by $(\overline{x}^1, \overline{u}^1)$.

$$\sum_{t-L^1+1}^{t} \overline{u}^1_t = \sum_{t-L^1+1}^{t} \sum_{t''=t'+1}^{T+1} f^1(t',t'')$$

As variables $f^1(t',t'') = 0$ if $t'' < t' + L^1$, it yields

$$\sum_{t'=t-L^1+1}^{t} \overline{u}_t^1 = \sum_{t-L^1+1}^{t} \sum_{t''=t+1}^{T+1} f^1(t',t'') \leq \overline{x}_t^1$$

The min-down constraint can be checked similarly. Note that inequality (1.4) is also satisfied:

$$\overline{x}_t^1 - \overline{x}_{t-1}^1 = \sum_{t''=t+1}^{T+1} f^1(t,t'') - \sum_{t'=1}^{t-1} f^1(t',t) \leq \overline{u}_t^1$$

The cost of solution $(\overline{x}^1, \overline{u}^1)$ is

$$\left(\sum_{t=1}^{T} c_f^1 \sum_{t'=1}^{t} \sum_{t''=t+1}^{T+1} f^1(t',t'')\right) + \left(\sum_{t=2}^{T} c_0^1 \sum_{t''=t+1}^{T+1} f^1(t,t'')\right)$$

which, by reindexing, is equal to

$$\left(\sum_{t'=1}^{T} \sum_{t''=2}^{T+1} \sum_{t'=t'}^{t''-1} c_f^1 f^1(t',t'')\right) + \left(\sum_{t=2}^{T} c_0^1 \sum_{t''=t+1}^{T+1} f^1(t,t'')\right)$$

This is exactly the cost of $(\overline{f}^1, \overline{g}^1)$, therefore $(\overline{f}^1, \overline{g}^1)$ and $(\overline{x}^1, \overline{u}^1)$ have same cost. This concludes the proof. ■

Now we prove the reverse inequality $v(F_{x,u}^1) \geq v(F^1\text{-}Flow)$. Consider a solution $(\overline{x}, \overline{u}) \in P_{x,u}^1$ and an arbitrary cost vector $c_{x,u}$. We use Algorithm 1 to construct a solution $(\overline{f}, \overline{g}) \in P_{flow}^1$ with same cost as $(\overline{x}, \overline{u})$. Recall bipartite graph G with vertices $V = V^U \cup V^D$, where $v_t^u \in V^U$ corresponds to a start-up of unit 1 at time t and $v_t^d \in V^D$ corresponds to a shut down at time t. The arc associated with flow variable $f^1(t,t')$ (resp. $g^1(t,t')$) connects the start-up (resp. shut down) at time t to the shut-down (resp. start-up) at time t'.

Algorithm 1 will construct a solution $(\overline{f}, \overline{g})$ such that $d(v_t^u)$ is the quantity of flow that must enter node v_t^u, and $d(v_t^d)$ is the quantity of flow that must exit node v_t^d, where:

$$\begin{aligned} d(v_t^u) &= \overline{u}_t \\ d(v_t^d) &= \overline{u}_t - (\overline{x}_t - \overline{x}_{t-1}) \end{aligned}$$

At each iteration of the algorithm, $\text{In}(v)$ and $\text{Out}(v)$ are updated, where $\text{In}(v)$ (resp. $\text{Out}(v)$) is the current quantity of flow entering (resp. exiting) node v. Algorithm 1 constructs solution $(\overline{f}, \overline{g})$ from $(\overline{x}, \overline{u})$.

Note that at any iteration t of the first loop of Algorithm 1, for any $k < t$ the following holds:

$$\begin{aligned} \textstyle\sum_{t'=1}^{k} \text{In}(v_{t'}^d) - \text{Out}(v_{t'}^d) &= 1 - \overline{x}_k \\ \textstyle\sum_{t'=1}^{k} \text{In}(v_{t'}^u) - \text{Out}(v_{t'}^u) &= \overline{x}_k \end{aligned}$$

In particular, $\sum_{t'=1}^{t-1} \text{In}(v_{t'}^d) - \text{Out}(v_{t'}^d) = 1 - \overline{x}_{t-1}$. As $\overline{u}_t \leq 1 - \overline{x}_{t-1}$ by min-down constraint (1.3), there is always enough incoming flow in nodes v_1^d, ..., v_{t-1}^d to convey $\overline{u}_t$ units of flow to v_t^u. Similarly,

Algorithm 1 Construction of solution $(\overline{f},\overline{g})$

$\text{In}(v) = \text{Out}(v) = 0$, for each node v, $(\overline{f},\overline{g}) = 0$
$\text{In}(v_1^u) = \overline{x}_1$, $\text{In}(v_1^d) = 1 - \overline{x}_1$
for $t = 2, ..., T$ **do**
  for $t' = 1, ..., t-1$ **do**
    if $\text{In}(v_{t'}^d) - \text{Out}(v_{t'}^d) > 0$ and $\text{In}(v_t^u) < d(v_t^u)$ **then**
      $\text{flow} = \min\left(d(v_t^u) - \text{In}(v_t^u),\quad \text{In}(v_{t'}^d) - \text{Out}(v_{t'}^d)\right)$
      $\overline{g}(t',t) \leftarrow \overline{g}(t',t) + \text{flow}$
      $\text{Out}(v_{t'}^d) \leftarrow \text{Out}(v_{t'}^d) + \text{flow}$
      $\text{In}(v_t^u) \leftarrow \text{In}(v_t^u) + \text{flow}$
    end if
    if $\text{In}(v_{t'}^u) - \text{Out}(v_{t'}^u) > 0$ and $\text{In}(v_t^d) < d(v_t^d)$ **then**
      $\text{flow} = \min\left(d(v_t^d) - \text{In}(v_t^d),\quad \text{In}(v_{t'}^u) - \text{Out}(v_{t'}^u)\right)$
      $\overline{f}(t',t) \leftarrow \overline{f}(t',t) + \text{flow}$
      $\text{Out}(v_{t'}^u) \leftarrow \text{Out}(v_{t'}^u) + \text{flow}$
      $\text{In}(v_t^d) \leftarrow \text{In}(v_t^d) + \text{flow}$
    end if
  end for
end for
for $t = 1, ..., T$ **do**
  $\overline{f}(t,T+1) = \text{In}(v_t^u) - \text{Out}(v_t^u), \quad \overline{g}(t,T+1) = \text{In}(v_t^d) - \text{Out}(v_t^d)$
end for

$\sum_{t'=1}^{t-1} \text{In}(v_{t'}^u) - \text{Out}(v_{t'}^u) = \overline{x}_{t-1}$, and $\overline{u}_t - (\overline{x}_t - \overline{x}_{t-1}) \le \overline{x}_t$, as $\overline{x}_t - \overline{x}_{t-1}$ is positive only if $\overline{u}_t = 0$. Thus, there is always enough incoming flow in nodes v_1^u, ..., v_{t-1}^u to convey $\overline{u}_t - (\overline{x}_t - \overline{x}_{t-1})$ units of flow to v_t^d.

Therefore, at any iteration t of the first loop of Algorithm 1, for any $k < t$:

$$\begin{aligned} \text{In}(v_k^d) &= d(v_k^d) \\ \text{In}(v_k^u) &= d(v_k^u) \end{aligned}$$

Thus, when Algorithm 1 terminates,

$$\begin{aligned} \overline{x}_t &= \sum_{t'=1}^{t} \sum_{t''=t+1}^{T+1} \overline{f}(t',t'') \quad \forall t \in \mathcal{T} \\ \overline{u}_t &= \sum_{t''=t+1}^{T+1} \overline{f}(t,t'') \quad \forall t \ge 2 \end{aligned}$$

As in the proof of Theorem 3.2, this proves that $(\overline{x},\overline{u})$ and $(\overline{f},\overline{g})$ have same cost.

As by construction, constraints (1.15), (1.16) and (1.17) are satisfied, there only remains to prove that the min-up and min-down time constraints are satisfied, *i.e.*, the flow from a node v_t^d to $v_{t'}^u$ (resp. v_t^u to $v_{t'}^d$) is zero if $t' < t + \ell^1$ (resp. $t' < t + L^1$).

Lemma 3.2. *Given solution $(\overline{f},\overline{g})$ constructed by Algorithm 1, the following holds:*

$$\begin{aligned} &(i) \quad \overline{g}(t,t') = 0 \quad \forall t \ge 2,\ \forall t' \in \{t+1, ..., t+\ell^1 - 1\} \\ &(ii) \quad \overline{f}(t,t') = 0 \quad \forall t \ge 2,\ \forall t' \in \{t+1, ..., t+L^1 - 1\} \end{aligned}$$

Proof. We prove (i) by contradiction; (ii) can be proved similarly.

Suppose there exist $t_0 > 0$ and $k_0 > 0$ such that $\overline{g}(t_0,k_0) > 0$ and $k_0 < t_0 + \ell^1$.

We will prove that $\overline{x}_{t_0-1} > 1 - \sum_{k=t_0}^{k_0} \overline{u}_k$. Let

$$\alpha = 1 - \overline{x}_{t_0-1} = \sum_{t=1}^{t_0-1} \sum_{k=t_0}^{T} \overline{g}(t,k)$$

By construction, for each $t \leq t_0 - 1$, $k > k_0$, $\overline{g}(t,k) = 0$, otherwise Algorithm 1 would not have assigned a positive flow $\overline{g}(t_0,k_0)$ from $v_{t_0}^d$ to $v_{k_0}^u$. Therefore

$$\alpha = \sum_{t=1}^{t_0-1} \sum_{k=t_0}^{k_0} g(t,k) = \left(\sum_{k=t_0}^{k_0-1} \sum_{t=1}^{t_0-1} \overline{g}(t,k) \right) + \sum_{t=1}^{t_0-1} \overline{g}(t,k_0)$$

As for each k, $\overline{u}_k = \sum_{t''=k+1}^{T+1} \overline{f}(t,t'') = \sum_{t'=1}^{k} \overline{g}(t',t')$,

$$\begin{array}{ll} \sum_{t=1}^{t_0-1} \overline{g}(t,k) \leq \overline{u}_k & \forall k \in \{t_0,\ldots,k_0-1\} \\ \sum_{t=1}^{t_0-1} \overline{g}(t,k_0) \leq \overline{u}_{k_0} - g(t_0,k_0) & \end{array}$$

Therefore $\alpha \leq \sum_{k=t_0}^{k_0} \overline{u}_k - \overline{g}(t_0,k_0)$, thus $\overline{x}_{t_0-1} > 1 - \sum_{k=t_0}^{k_0} \overline{u}_k$.

It follows that inequality (1.3) is not satisfied by $(\overline{x},\overline{u})$ at time $t = t_0 + \ell - 1$,which is a contradiction. ■

Finally, we have proved the following.

Theorem 3.3. $v(F^1_{x,u}) \geq v(F^1\text{-}Flow)$

As this result holds for any cost vector c, the integrality of $F^1_{x,u}$ is obtained from the integrality of (F^1-*Flow*) by Theorem 1.1. It thus gives another proof of Theorem 1.5.

3.2 Polyhedral study

The MUCP polytope is denoted by P^n_{UCP}:

$$P^n_{UCP} = conv\Big\{ (x,u,p) \text{ satisfying (1.2) - (1.8)} \Big\}$$

In this section, we give some first polyhedral results on polytope P^n_{UCP}, for any number n of units. In order to study the combinatorial structure of P^n_{UCP}, its projection $P^n_{x,u}$ on binary variables x and u is considered in the following lemma.

Lemma 3.3. *The projection of polytope P^n_{UCP} on variables (x,u) is*

$$P^n_{x,u} = Conv\Big\{ (x,u) \in \{0,1\}^{\mathcal{N}\times\mathcal{T}} \times \{0,1\}^{\mathcal{N}\times\mathcal{T}\setminus\{1\}} \text{ s. t. (1.2), (1.3), (1.4) and } \sum_{i=1}^{n} P^i_{max} x^i_t \geq D_t \ \forall t \in \mathcal{T} \Big\}$$

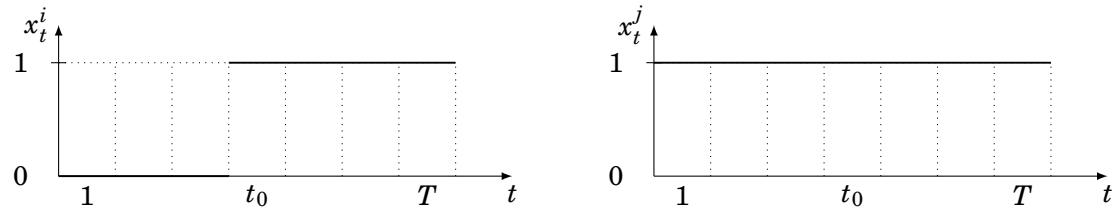
Proof. Let $Proj_{x,u}(P^n_{UCP})$ be the projection of polytope P^n_{UCP} on variables (x,u). Given $(\tilde{x},\tilde{u}) \in P^n_{x,u}$, set $\tilde{p}^i_t = P^i_{max}\tilde{x}^i_t$, $\forall i \in \mathcal{N}$, $\forall t \in \mathcal{T}$, then $(\tilde{x},\tilde{u},\tilde{p}) \in P^n_{UCP}$. Conversely, consider $(\overline{x},\overline{u}) \in Proj_{x,u}(P^n_{UCP})$. There exists $\overline{p}$ such that $(\overline{x},\overline{u},\overline{p}) \in P^n_{UCP}$. Thus $\sum_{i=1}^n \overline{p}^i_t \geq D_t$ $\forall t \in \mathcal{T}$ and $\overline{p}^i_t \leq P^i_{max}\overline{x}^i_t$. It follows that $\sum_{i=1}^n P^i_{max}\overline{x}^i_t \geq D_t$, and thus $(\overline{x},\overline{u}) \in P^n_{x,u}$. ■

We first introduce some vectors that will be useful in the polyhedral proofs. Note that a solution of $P^n_{x,u}$ is a couple $(x,u) \in \{0,1\}^{\mathcal{N}\times\mathcal{T}} \times \{0,1\}^{\mathcal{N}\times\mathcal{T}\setminus\{1\}}$ resulting in $n(2T-1)$ coordinates. Given a unit $i \in \mathcal{N}$ and a time period $t \in \mathcal{T}$, let $\chi^u_{i,t}$ (resp. $\chi^d_{i,t}$) be the vector such that unit i is down (resp. up) on $[1,t-1]$, starts up (resp. shuts down) at time t and remains up (resp. down) on $[t,T]$, and such that unit j is up at all times, for all $j \neq i$. Moreover, let $\chi_0 \in P^n_{x,u}$ be the vector in which all units are up at all times. To illustrate, the coordinates of vector χ^u_{i,t_0} are the following:

$$\begin{array}{rccccccc} & 1 & \cdots & & t_0 & & \cdots & T \\ x^i = & \big(0, & ..., & 0, & 1, & 1, & ..., & 1\big) \end{array} \quad \text{and} \quad x^j = (1, \ ..., \ 1) \quad j \neq i$$

$$\begin{array}{rccccccc} & 1 & \cdots & & t_0 & & \cdots & T \\ u^i = & \big(0, & ..., & 0, & 1, & 0, & ..., & 0\big) \end{array} \quad \text{and} \quad u^j = (0, \ ..., \ 0) \quad j \neq i$$

A simple way to present the x coordinates of vector χ^u_{i,t_0}, $t_0 \in \mathcal{T}\setminus\{1\}$, is the diagram of Figure 3.1.



(a) Coordinates x^i (b) Coordinates x^j, for $j \neq i$

Figure 3.1: Coordinates x of vector χ^u_{i,t_0}

The proofs of the two following theorems are extensions of results for the 1-unit polytope from [79].

Theorem 3.4. *The polytope $P^n_{x,u}$ is full-dimensional if and only if for all $i \in \mathcal{N}$, $\sum_{j\in\mathcal{N}\setminus\{i\}} P^j_{max} \geq \max_{t\in\mathcal{T}} D_t$.*

Proof. First if there is a unit i such that $\sum_{j\in\mathcal{N}\setminus\{i\}} P^j_{max} < \max_{t\in\mathcal{T}} D_t$, then there exists a time t such that $\sum_{j\in\mathcal{N}\setminus\{i\}} P^j_{max} < D_t$. There is no solution (x,u) such that $x^i_t = 0$, thus $P^n_{x,u}$ is not full-dimensional.

Now suppose the hypothesis holds, *i.e.* at each time t, the units of $\mathcal{N}\setminus\{i\}$ are sufficient to cover the demand, $\forall i \in \mathcal{N}$. Thus, given a unit $i \in \mathcal{N}$, vectors $(\chi^d_{i,t},\ t \in \mathcal{T})$ and vectors $(\chi^u_{i,t},\ t \in \mathcal{T}\setminus\{1\})$ are $2T-1$ incident vectors of solutions in $P^n_{UCP}(x,u)$. Hence, vectors $\chi^u_{i,t}$, $\chi^d_{i,t}$ and vectors χ_0 constitute

a set of $n(2T-1)+1$ affinely independent vectors of $P^n_{x,u}$. It follows that $P^n_{x,u}$ is full-dimensional. ■

In the following, we will consider the full-dimensionality condition is satisfied, *i.e.* $n-1$ units or less are always sufficient to meet the demand at any time. Note that this assumption is required to come up with a reliable production plan.

Theorem 3.5. *Inequalities (1.3), (1.4) and trivial inequalities $u^i_t \geq 0$, $i \in \mathcal{N}$, $\forall t \in \mathcal{T}\setminus\{1\}$, describe facets of $P^n_{x,u}$.*

Proof. Let $i_0 \in \mathcal{N}$ and $t_0 \in \mathcal{T}\setminus\{1\}$.

Vectors $(\chi^u_{i,t},(i,t) \in (\mathcal{N}\times\mathcal{T}\setminus\{1\}) \setminus \{(i_0,t_0)\})$, vectors $(\chi^d_{i,t},(i,t) \in \mathcal{N}\times\mathcal{T})$, and vector χ_0 are $n(2T-1)$ affinely independent vectors of $P^n_{x,u}$ satisfying $u^{i_0}_{t_0} = 0$. So the trivial inequality defines a facet of $P^n_{x,u}$.

Vectors $(\chi^u_{i,t},(i,t) \in \mathcal{N}\times\mathcal{T}\setminus\{1\})$, vectors $(\chi^d_{i,t},(i,t) \in (\mathcal{N}\times\mathcal{T})\setminus\{(i_0,t_0)\})$ and vector χ_0 are $n(2T-1)$ affinely independent vectors of $P^n_{x,u}$ satisfying $u^{i_0}_{t_0} = x^{i_0}_{t_0} - x^{i_0}_{t_0-1}$. So (1.4) defines a facet of $P^n_{x,u}$.

As inequality (1.3) has been proven to be facet defining for the 1-unit polytope $P^1_{x,u}$ (see [79]), there exist $2T-1$ affinely independent vectors $(x^i,u^i) \in P^1_{x,u}$ satisfying $\sum_{t=t_0-\ell^{i_0}+1}^{t_0} u^{i_0}_t = 1 - x^{i_0}_{t_0-\ell^{i_0}}$. From each vector (x^{i_0},u^{i_0}) we construct a vector $(\overline{x},\overline{u}) \in P^n_{x,u}$ satisfying $\sum_{t=t_0-\ell^{i_0}+1}^{t_0} \overline{u}^{i_0}_t = 1 - \overline{x}^{i_0}_{t_0-\ell^{i_0}}$, by setting coordinates as follows: $(\overline{x}^j_t = 1,\ j \neq i,\ t \in \mathcal{T})$, $(\overline{u}^j_t = 0,\ j \neq i,\ t \in \mathcal{T}\setminus\{1\})$, and $\overline{x}^i_t = x^i_t$, $\overline{u}^i_t = u^i_t$, $\forall t$. These $2T-1$ vectors of $P^n_{x,u}$ alongside with the $(n-1)(2T-1)$ vectors $(\chi^u_{j,t},\ j \neq i,\ t \in \mathcal{T}\setminus\{1\})$, $(\chi^d_{j,t},\ j \neq i,\ t \in \mathcal{T})$, constitute a set of $n(2T-1)$ affinely independent vectors of $P^n_{x,u}$ satisfying inequality (1.3) with equality, which proves that (1.3) defines a facet of $P^n_{x,u}$. ■

Property $\Pi_{i,t}$ Given a face F of $P^n_{x,u}$, a unit i and a time period t, Property $\Pi_{i,t}$ is as follows:

$$\text{Solution } (x,u) \in F \text{ satisfies } \Pi_{i,t} \iff \text{Unit } i \text{ is down on } [t,t+\ell^i] \text{ } i.e.\ x^i_{t'} = 0,\ \forall t' \in [t,t+\ell^i].$$

Let us consider the transformation $\Psi^i_{t_0,t_1}$ such that for any vector $\rho \in \{0,1\}^{\mathcal{N}\times\mathcal{T}} \times \{0,1\}^{\mathcal{N}\times\mathcal{T}\setminus\{1\}}$, $\Psi^i_{t_0,t_1}(\rho)$ is equal to vector ρ except for unit i which is down over $[t_0,t_1]$ and up the rest of the time.

We give a generic technical lemma.

Lemma 3.4. *Let $\sum_{j\in\mathcal{N}} a^j x^j + \sum_{j\in\mathcal{N}} b^j u^j \leq \gamma$ be a valid inequality for $P^n_{x,u}$, different from inequality (1.3). Let F be the associated face.*

(*i*) *If F is a facet, then for all $i \in \mathcal{N}$ and $t \in \mathcal{T}$, there exists $(x,u) \in F$ satisfying property $\Pi_{i,t}$.*

(*ii*) *For a given $i \in \mathcal{N}$, if for all $t \in \mathcal{T}$ there exists $(x,u)_t \in F$ satisfying property $\Pi_{i,t}$, and if neither variables x^i nor u^i appear in $\sum_{j\in\mathcal{N}} a^j x^j + \sum_{j\in\mathcal{N}} b^j u^j$ (i.e. $a^i = b^i = 0$), then for all $t \in \mathcal{T}$, F contains the following solutions:*

- *Solution* $\Psi^i_{t,t+\ell^i}((x,u)_t)$*, where unit* i *is up on* $[1,t-1]$*, down on* $[t,t+\ell^i]$ *and up on* $[t+\ell^i+1,T]$.
- *Solution* $\Psi^i_{t+1,t+\ell^i}((x,u)_t)$*, where unit* i *is up on* $[1,t]$*, down on* $[t+1,t+\ell^i]$ *and up on* $[t+\ell^i+1,T]$.
- *Solution* $\Psi^i_{t,t+\ell^i-1}((x,u)_t)$*, where unit* i *is up on* $[1,t-1]$*, down on* $[t,t+\ell^i-1]$ *and up on* $[t+\ell^i,T]$.
- *Solution* $\Psi^i_{1,t_0}((x,u)_1)$*, for any* $t_0\in[1,\ell^i-1]$*, where unit* i *is down on* $[1,t_0]$ *and up on* $[t_0+1,T]$.

Proof. (i): Suppose $\Pi_{i,t}$ does not hold for given $i_0\in\mathcal{N}$ and $t_0\in\mathcal{T}$. Thus, for any given solution $(\tilde{x},\tilde{u})\in F$, if unit i_0 is down at time t_0, it must start up before time $t_0+\ell^{i_0}$ (if $t_0>T-\ell^{i_0}$ we can consider $t_0=T-\ell^{i_0}$ w.l.o.g.). Then $(\tilde{x},\tilde{u})$ satisfies $\sum_{t=t_0+1}^{t_0+\ell^{i_0}}u_t^{i_0}\geq 1-x_{t_0}^{i_0}$. As inequality (1.3) holds too, it follows that $\sum_{t=t_0+1}^{t_0+\ell^{i_0}}\tilde{u}_t^{i_0}=1-\tilde{x}_{t_0}^{i_0}$. Thus F is included in the face of inequality (1.3), which contradicts the fact that F is different from the face defined by (1.3).

(ii): For each $t\in\mathcal{T}$, from Property $\Pi_{i,t}$, $\exists(x,u)_t\in F$ such that $x^i_{t'}=0\ \forall t'\in[t,t+\ell^i]$. Then vector $\Psi^i_{t,t+\ell^i}((x,u)_t)$ is still a solution as unit i remains down for ℓ^i+1 periods, thus satisfying the min-down constraints. Moreover, the demand is satisfied since unit i is up in vector $\Psi^i_{t,t+\ell^i}((x,u)_t)$ at least as often as in solution (x,u). As $a^i=b^i=0$, solution $\Psi^i_{t,t+\ell^i}((x,u)_t)$ is a solution of F. Similarly, vectors $\Psi^i_{t+1,t+\ell^i}((x,u)_t)$, $\Psi^i_{t,t+\ell^i-1}((x,u)_t)$ and $\Psi^i_{1,t_0}((x,u)_1)$, for any $t_0\in[1,\ell^i-1]$ are solutions of F. ■

We now prove that min-up inequalities (1.2) are facet defining under a mild condition.

Theorem 3.6 (Facet defining min-up inequalities)**.** *For* $i\in\mathcal{N}$*, for* $t_1\in\{L^i+1,\dots,T\}$*, let* F *be the face defined by the min-up inequality (1.2)* $\sum_{t=t_1-L^i+1}^{t_1}u_t^i\leq x_{t_1}^i$*.* F *is a facet of* $P^n_{x,u}$ *if and only if for any unit* $j\in\mathcal{N}\backslash\{i\}$ *and time* $t\in\mathcal{T}$*, there exists a solution* $(x,u)\in F$ *satisfying* $\Pi_{i,t}$*.*

Proof. The necessity ($\Longrightarrow$) follows from Lemma 3.4 (i).

We prove the sufficiency ($\Longleftarrow$). Suppose that F is included in the face of an inequality $\sum_{j\in\mathcal{N}}\left(\sum_{t\in\mathcal{T}}a_t^jx_t^j+\sum_{t\in\mathcal{T}\backslash\{1\}}b_t^ju_t^j\right)\leq\gamma$. The claim is that:
$F=\{(x,u)\in P^n_{x,u}\,|\,\sum_{j\in\mathcal{N}}\left(\sum_{t\in\mathcal{T}}a_t^jx_t^j+\sum_{t\in\mathcal{T}\backslash\{1\}}b_t^ju_t^j\right)=\gamma\}$, which proves that F is a facet of $P^n_{x,u}$.

For any $j\in\mathcal{N}\backslash\{i\}$, there are no x^j nor u^j variables appearing in inequality $\sum_{t=t_1-L^i+1}^{t_1}u_t^i\leq x_{t_1}^i$. Since there exists $(x,u)_t\in F$ satisfying $\Pi_{j,t}$ for any $t\in\mathcal{T}$, it follows from Lemma 3.4 (ii) that for any $t\in\mathcal{T}$, $\Psi^i_{t,t+\ell^j}((x,u)_t)\in F$ and $\Psi^i_{t+1,t+\ell^j}((x,u)_t)\in F$. As these solutions differ only over variable x_t^i, we can conclude $a_t^j=0$, $\forall j\in\mathcal{N}\backslash\{i\}$, $\forall t\in\mathcal{T}$. Moreover, Lemma 3.4 (ii) implies that $\Psi^i_{T-\ell^i,T}((x,u)_t)\in F$, which differs from $\Psi^i_{T-\ell^i,T-1}((x,u)_t)\in F$ only over x_T^i and u_T^i variables. As $a_T^i=0$, it follows that $b_T^i=0$. Similarly we can see that $b_t^j=b_{t-1}^j$, $\forall j\in\mathcal{N}\backslash\{i\}$, $\forall t\in\mathcal{T}\backslash\{1,2\}$, by comparing vector $\Psi^i_{t,t+\ell^i-1}((x,u)_t)\in F$, $\forall t\in\mathcal{T}$ to vector $\Psi^i_{t,t+\ell^i}((x,u)_t)\in F$, $\forall t\in\mathcal{T}$, and vector

$\Psi^i_{1,t'}((x,u)_1) \in F$, $t' \in [1, \ell^j - 1]$, to vector $\Psi^i_{1,1+\ell^i}((x,u)_1) \in F$. It follows that $b^j_t = 0$, $\forall j \in \mathcal{N}\backslash\{i\}$, $\forall t \in \mathcal{T}\backslash\{1\}$.

Vectors $\chi^d_{i,t_1} \in F$ and $\chi^d_{i,t_1-1} \in F$ differ only over variable $x^i_{t_1-1}$. Thus $a^i_{t_1-1} = 0$. Then, from vectors $\chi^d_{i,t} \in F$, $t \in [1,...,t_1]$, we can iteratively see that $a^i_t = 0$ for all $t \leq t_1 - 1$. We introduce vectors $\Theta^i_{t,t'} \in P^n_{x,u}$ such that i starts up at time t, stays up until t' and shuts down at time $t'+1$ (all other units are up at all times). As $P^n_{x,u}$ is full-dimensional, note that $n-1$ units are always sufficient to meet the demand at any time, thus for any $t > t_1$, vectors $\Theta^i_{t_1-L^i+1,t} \in F$. Moreover, vectors $\Theta^i_{t_1-L^i+1,t-1}$ and $\Theta^i_{t_1-L^i+1,t}$ differ only over variable x^i_t, which implies $a^i_t = 0$ for any $t > t_1$. Moreover, vector $\chi^d_{i,1} \in F$ and for any $t \geq t_1 - L^i + 1$, vector $\chi^u_{i,t} \in F$. Since vector $\chi^u_{i,t}$ differs from vector $\chi^d_{i,1}$ only over variables $x^i_{t'}$, $t' \geq t$ and variable u^i_t, $b^i_t = -a^i_{t_1}$ for any $t \in [t_1 - L^i + 1, t_1]$, and $b^i_t = 0$ for any $t > t_1$. Finally, for any $t \leq t_1 - L^i$, vectors $\Theta^i_{t,t_1-1} \in F$, and Θ^i_{t,t_1-1} differs from $\chi^d_{i,1}$ only over variables $x^i_{t'}$, $t' \in [t, t_1 - 1]$ and variable u^i_t, it follows that for any $t \leq t_1 - L^i$, $b^i_t = 0$.

The remaining inequality is then:

$$-a^i_{t_1} x^i_{t_1} + \sum_{t=t_1-L^i+1}^{t_1} a^i_{t_1} u^i_t = \gamma.$$

Since $\chi^d_{i,1} \in F$, $\gamma = 0$, which proves that F is a facet of $P^n_{x,u}$. ■

3.3 Rank of unit subsets

In order to introduce new valid inequalities for the MUCP polytope, we define the rank of a unit subset, which captures both dynamic and knapsack aspects of the MUCP.

Rank For each subset of units $M \subset \mathcal{N}$, its *rank* $\alpha_t(M)$ is the smallest number of units that must be up in M at time t in a feasible solution.

Since this rank is hard to compute, a static version is also considered as it will be useful in practice.

Static rank For each subset of units $M \subset \mathcal{N}$, its *static rank* is the smallest number of units that must be up in M at time t in order to satisfy the residual demand $D_t - \sum_{j \notin M} P^j_{max}$.

As all feasible solutions meet the demand at time t, it is clear that:

$$\alpha_t(M) \geq \overline{\alpha}_t(M) \ \ \forall t \in \mathcal{T} \ \forall M \subset \mathcal{N}.$$

Example 3.1. *Recall the illustrative instance of the MUCP from Example 1.1, with* $T = 3$, $D = [20, 10, 25]$ *and three units such that* $P^1_{max} = 15$, $P^2_{max} = 5$, , $P^3_{max} = 5$ *and* $\ell^1 = \ell^2 = \ell^3 = 2$, $L^1 = L^2 = L^3 = 2$.

Let $M = \{1,3\}$. *Even if unit* 2 *is up, its production alone is not sufficient to satisfy the demand at time 1. Subtracting production of unit* 2, *the residual demand to be satisfied is* $\overline{D}_1 = D_1 - P^2_{max} = 15$. *One unit in* M *must then be up to cover the demand. Thus* $\overline{\alpha}_1(M) \geq 1$. *Since only one unit of* M *is*

enough to cover the residual demand $\overline{D}_1$ (here unit 1 is enough), $\overline{\alpha}_1(M) = 1$. Then $\alpha_1(M) = 1$, as there exists a feasible solution in which only one unit of M is up at time 1, for example the solution, illustrated by Figure 3.2a, in which unit 1 and 2 are up at all times and unit 3 is down at times 1 and 2 and up at time 3. The demand is represented with dotted lines.

Let us now consider $M' = \{1,2,3\}$. The static rank of M' at time 2 is equal to 1: $\overline{\alpha}_2(M') = 1$, since one unit of M' is necessary and sufficient to satisfy the residual demand $\overline{D}_2 = D_2 = 5$. Indeed, if there were no min-up / min-down constraints, the solution given by Figure 3.2b would be feasible.

However, in this example, min-down constraints hold and therefore $\overline{\alpha}_2(M') \neq \alpha_2(M')$. Solution in Figure 3.2b does not satisfy min-down constraint, and there is actually no feasible solution in which only one unit of M' is up at time 2. First note $\overline{\alpha}_1(M') = 2$ and $\overline{\alpha}_1(M') = 3$, i.e., as at least two (resp. three) units of M' must be up up at times 1 (resp. 3). Let us now assume there exists a feasible solution in which only one unit of M' is up at time 2. Therefore, one unit of M' must shut down at time 2 and start up at time 3. Since the minimum down time of each unit of M' is 2, this leads to a contradiction. Therefore $\alpha_2(M') > 1$. As Figure 3.2a gives a feasible solution with two units of M' up at time 2, $\alpha_2(M') = 2$.

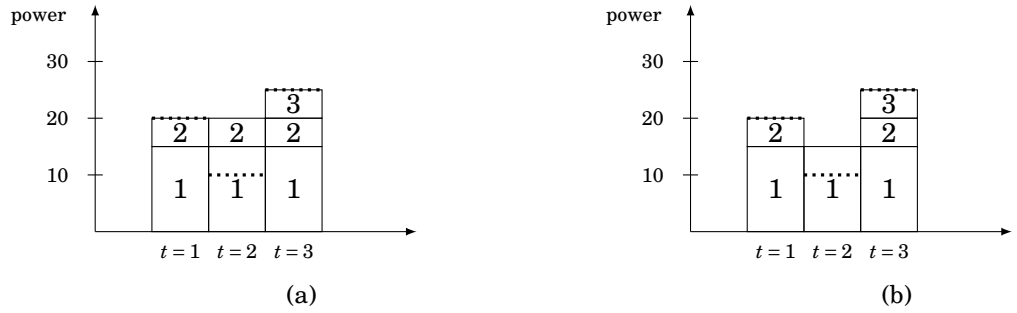


Figure 3.2

Note that computing the rank of a unit subset is an optimization problem. It is to find the smallest number of units that must be up in M at time t in a feasible solution. In order to state the problem's complexity, let us consider its decision version: given an instance of the MUCP, a time period t_0, a unit subset M and an integer K, the question is whether there exists a feasible solution in which at most K units of M are up at time t_0, *i.e.* $\alpha_{t_0}(M) \leq K$.

Theorem 3.7. *Computing the rank of a unit subset is NP-hard for $T \geq 3$.*

Proof. Let us consider an instance of the partition problem, with a set of n positive integers $a_1, ..., a_n$. The question is whether $S = \{1, ..., n\}$ can be partitioned into two subsets S_1 and S_2 such that $\sum_{i \in S_1} a_i = \sum_{i \in S_2} a_i$.

First note that if such a partition exists, then $\sum_{i \in S_1} a_i = \sum_{i \in S_2} a_i = A$ where $A = \frac{1}{2} \sum_{i \in S} a_i$. Consider now the following instance of the rank decision problem: let $T = 3$ with $D = [A, 0, A]$,

and n units such that $P^i_{max} = P^i_{min} = a^i$ and $\ell^i = 2$, $i \in \{1,...,n\}$. The other characteristics are fixed arbitrarily. Set $t_0 = 2$, $M = \{1,...,n\}$ and $K = 0$. Let us suppose there exists a solution to the latter instance. Let S_1 be the set of units up at time 1, and S_2 be the set of units up at time 3. The claim is that (S_1, S_2) is a solution to the partition problem. Indeed, S_1 and S_2 are disjoint, as all units up at time 1 shut down at time 2, and stay down for a minimum of two time periods. Thus, all units up at time 1 are down at time 3. Moreover, the units in S_1 satisfy the demand at time 1, so $\sum_{i\in S_1} a_i \geq A$. Similarly, $\sum_{i\in S_2} a_i \geq A$. As S_1 and S_2 are disjoint, $2A \leq \sum_{i\in S_1} a_i + \sum_{i\in S_2} a_i \leq \sum_{i\in S} a_i = 2A$, it follows $\sum_{i\in S_1} a_i = A$, $\sum_{i\in S_2} a_i = A$ and $S_1 \cup S_2 = S$.

Conversely, any solution to the partition problem can similarly be used to construct a solution to this instance of the rank computation problem. ■

Given this complexity result, the static rank will be used in practice instead of the rank. Indeed, the static rank can be computed in linear time (provided the units are sorted by decreasing order according to P^j_{max}) using Algorithm 2.

Algorithm 2 Computation of the static rank of set M at time t

```
Compute the residual demand at time t: D̄_t = D_t − Σ_{j∉M} P^j_max
Sort units in M by decreasing order according to P_max
ρ = 0 and α = 0
while ρ < D̄_t do
    ρ ← ρ + M[α]
    α ← α + 1
end while
return α
```

For a given subset of units M, the definition of the rank at a given time t is extended to a given interval $\mathscr{I} = \{t_0, ..., t_1\}$. The maximum rank of M over $\mathscr{I}$ is denoted by $\alpha_{\mathscr{I}}(M)$, *i.e.* $\alpha_{\mathscr{I}}(M) = \max_{t\in\mathscr{I}} \alpha_t(M)$. Similarly, the definition of the static rank at a given time t is extended to $\mathscr{I}$. The maximum static rank of M over $\mathscr{I}$ is denoted by $\overline{\alpha}_{\mathscr{I}}(M)$.

Let t_{max} be the time period at which the demand is maximum on $\mathscr{I}$.

Lemma 3.5. $\alpha_{t_{max}}(M) = \alpha_{\mathscr{I}}(M)$.

Proof. By definition of the rank of M at time t_{max}, there exists a solution $(x,u) \in P^n_{x,u}$ such that exactly $\alpha_{t_{max}}(M)$ units in M are up at time t_{max}. Let M_{max} be the set of units in M which are up at time t_{max} in solution (x,u). Using (x,u), a solution $(\tilde{x},\tilde{u})$ is iteratively constructed such that any unit in $M\backslash M_{max}$ is down on the whole interval $\mathscr{I}$ and any unit in M_{max} is up on the whole interval $\mathscr{I}$. We first set $(\tilde{x},\tilde{u})$ equal to (x,u), and we slightly modify the behavior of the units in M as follows.

- For any $j \in M_{max}$, $\forall t \in \mathscr{T}$, we set $\tilde{x}^j_t = 1$.

- For any $j \in M \backslash M_{max}$, if j starts up at $t \in \mathscr{I}$ in solution (x,u), we update coordinates $(\tilde{x}^j, \tilde{u}^j)$ of $(\tilde{x},\tilde{u})$ such that j is down from time t to t_1, starts up at time t_1+1 and remains up until time T. Indeed, as the units in subset M_{max} meet the maximum demand $D_{t_{max}}$ on $\mathscr{I}$, for any $t \in \mathscr{I}$, the demand at time t is also met by units in M_{max}. For any $t \notin \mathscr{I}$, the units up at time t in $(\tilde{x},\tilde{u})$ are exactly the units up at time t in solution (x,u). Thus, in solution $(\tilde{x},\tilde{u})$, the demand is indeed satisfied at time t. Furthermore, by delaying the start-up of unit j, its minimum down time is satisfied, and since j remains up until the end of the time horizon, its minimum up time is satisfied as well.

- Similarly, for any $j \in M \backslash M_{max}$ such that j shuts down at time $t \in \mathscr{I}$, we set $(\tilde{x}^j, \tilde{u}^j)$ such that j is up from time 1 to $t_0 - 1$ and shuts down at time t_0. Similar arguments can prove that $(\tilde{x},\tilde{u})$ remains feasible.

Consequently, $(\tilde{x},\tilde{u})$ is a solution such that any unit in $M \backslash M_{max}$ is down on the whole interval $\mathscr{I}$ and any unit in C_{max} is up on the whole interval $\mathscr{I}$.

It follows that $\alpha_t(M) \leq \alpha_{t_{max}}(M), \forall t \in \mathscr{I}$. ■

Let, for any $i \in M$, the *static i-rank* $\overline{\alpha}^i_t(M)$ be the smallest number of units that must be up in M at time t in order to satisfy the residual demand $D_t - \sum_{j \notin M} P^j_{max}$, given that unit i is down at t. By Theorem 3.4, if $P^n_{UCP}(x,u)$ is full dimensional then the definition of $\overline{\alpha}^i_t(M)$ makes sense, as there exists a solution in which unit i is down at time t.

For example, by referring to the instance previously presented in this section, recall unit subset $M' = \{1,2,3\}$. The residual demand at time 2 is $\overline{D}_2 = D_2$. Unit 1 is sufficient to cover the residual demand, thus $\overline{\alpha}_2(M') = 1$. However, if unit 1 is down at time 2, there must be at least two units of M' up at time 2 to satisfy the residual demand. It follows that $\overline{\alpha}^1_2(M') = 2$.

3.4 Valid inequalities

In this section, we first define the up-set inequalities, which account for some of the combinatorial aspects induced by the knapsack structure of the MUCP. We will then introduce the interval up-set inequalities, as a generalization of the up-set inequalities. As they capture both knapsack constraints and minimum up and down times, they are more dedicated to the MUCP.

3.4.1 Up-set inequalities

By definition of the rank, for any subset $M \subset \mathscr{N}$ and time $t \in \mathscr{T}$, the *up-set inequality*, defined as follows, is valid:

$$\sum_{j \in M} x^j_t \geq \alpha_t(M). \tag{3.1}$$

This inequality is difficult to produce given that the rank $\alpha_t(M)$ is hard to compute. Thus, a weaker version of inequality (3.1) is also defined as the following *static up-set inequality*:

$$\sum_{j \in M} x_t^j \geq \overline{\alpha}_t(M). \tag{3.2}$$

In practice, if a lower bound α of $\overline{\alpha}_t(M)$ such that $\alpha \leq \alpha_t(M)$ is known, the corresponding inequality $\sum_{j \in M} x_t^j \geq \alpha$ can be used instead of (3.2).

These static up-set inequalities directly correspond to the extended cover inequalities for the knapsack polytope [4].

Facet-defining cases In [4], a characterization of the cases when these inequalities are facet-defining is given. These results are transposed to the MUCP. We first give a few definitions.

Let $\mathscr{P}^n = Conv\{x_t \in \{0,1\}^n, \sum_{j \in \mathscr{N}} P_{max}^j x_t^j \geq D_t\}$ be the polytope of the MUCP considered at time period t only.

Let $\mathscr{N}$ and all of its subsets to be considered below be ordered so that $P_{max}^j \geq P_{max}^{j+1}$, $j \in \{1,...,n-1\}$. As we place ourselves here at a given time period $t \in \mathscr{T}$, for simplicity, we drop the index t from all variables and quantities.

For a given $t \in \mathscr{T}$, a subset C of $\mathscr{N}$ is called an *up-set* if $\overline{\alpha}_t(C) \geq 1$. In other words, C is an up-set if and only if the units in $\mathscr{N} \backslash C$ are not sufficient to meet the demand at time t.

An up-set C is called *minimal* if for all subset $Q \subsetneq C$, $\overline{\alpha}_t(Q) = 0$. For any minimal up-set C, we define $E(C) = C \cup C'$ as the *extension of C to* $\mathscr{N}$, where

$$C' = \{j \in \mathscr{N} \backslash C, P_{max}^j \geq P_{max}^{j_1}\} \text{ where } j_1 = \arg\max_{j \in C} P_{max}^j.$$

A minimal up-set C is called *strong* if for any minimal up-set A such that $|A| = |C|$ and $A \neq C$, $E(C) \not\subset E(A)$. For example, by referring to the MUCP instance defined in Section 3.3, if we consider the time period $t = 2$, subset $M = \{1,3\}$ is a minimal up-set as neither $\{1\}$ nor $\{3\}$ is an up-set. Since M contains the most powerful unit of $\mathscr{N}$ (unit 1), $E(M) = M$. However, subset M is not strong. Indeed, subset $A = \{2,3\}$ is a minimal up-set such that $|A| = |M|$, and $E(A) = \{1,2,3\}$ implying that $E(M) \subset E(A)$.

A up-set M is said to be a *strong up-set extension* if there exists a strong up-set C such that:

(i) $M = E(C)$

(ii) $|C| = |M| - \overline{\alpha}_t(M) + 1$

(iii) $\overline{\alpha}_t(A) = 0$ where $A = C \backslash \{j_1, j_2\} \cup \{1\}$ and $j_2 = \arg\max_{j \in C \backslash \{j_1\}} P_{max}^j$.

For any subset $M \subset \mathscr{N}$, we denote by Up-Set(M) the corresponding inequality (3.2).

Before characterizing facet-defining cases, we give a few technical lemmas. These are transposed from [4].

Lemma 3.6. *For any up-set $C \in \mathcal{N}$:*

$$\overline{\alpha}(E(C)) \geq |E(C)\backslash C| + 1.$$

Furthermore, if C is minimal, then $\overline{\alpha}(E(C)) = |E(C)\backslash C| + 1$.

Proof. Consider a vector $\overline{x}$ such that $\sum_{j\in E(C)} \overline{x}^j \leq |E(C)\backslash C| = p$. In order to maximize the power production in solution $\overline{x}$, we need the p most powerful units in $E(C)$ to be up. The other units in $E(C)$ will be down, as $\sum_{j\in E(C)} \overline{x}^j \leq p$. Thus, the units up in $E(C)$ are exactly those in $|E(C)\backslash C|$. But as C is an up-set, the demand will not be met. So there is no vector $\overline{x}$ such that $\sum_{j\in E(C)} \overline{x}^j \leq |E(C)\backslash C|$ in $\mathscr{P}^n$. ■

Lemma 3.7. *A minimal up-set C is strong if and only if the set $R = C\backslash\{j_1\} \cup \{i_1\}$ is not an up-set, where i_1 is the most powerful unit of $\mathcal{N}\backslash E(C)$.*

Proof. ($\Rightarrow$) If R is an up-set, then, following from the minimality of C, R is also minimal. Furthermore, $|R| = |C|$, $R \neq C$ and $E(C) \subset E(R)$ since $i_1 \geq j_1$. So C is not strong.

($\Leftarrow$) If C is not strong, there exists a minimal up-set A such that $|A| = |C|$, $A \neq C$ and $E(C) \subset E(A)$. We can write $A = (C\backslash C_A) \cup A_C$, where $C_A = C\backslash A$ and $A_C = A\backslash C$. It follows from $|A| = |C|$ and $E(C) \subset E(A)$ that $|C_A| = |A_C|$ and $j_1 \in C_A$.

A is an up-set so $\overline{A} = \overline{C} \cup C_A \backslash A_C$ does not suffice to meet the demand by itself. As every unit in C_A is more powerful than each unit in A_C (or else we would not have $E(C) \subset E(A)$), we can deduce that $\overline{C} \cup \{j_1\}\backslash\{i_1\}$ does not suffice either to meet the demand. So R is an up-set. ■

For any up-set $M \subset \mathcal{N}$, we define $E^{-1}(M)$ the set of the $|M| - \overline{\alpha}(M) + 1$ less powerful units of M.

Lemma 3.8. *For any up-set $M \subset \mathcal{N}$, M is a strong up-set extension if and only if $E^{-1}(M)$ is a strong up-set which satisfies (iii) and such that $E(E^{-1}(M)) = M$.*

Proof. The set of the $|M| - \overline{\alpha}(M) + 1$ less powerful units of M is the only set that can possibly satisfy (i), (ii) and (iii). Hence the direct implication. The return implication follows from the definition of a strong up-set extension. ■

Lemma 3.9. *Let $M \subset \mathcal{N}$ be an up-set. If $E^{-1}(M)$ is an up-set, then either $M = E(E^{-1}(M))$ (and in that case $\sum_{j\in M} x^j \geq \overline{\alpha}(M)$ is exactly Up-Set($E(E^{-1}(M))$)) or the inequality $\sum_{j\in M} x^j \geq \overline{\alpha}(M)$ is dominated by Up-Set($E(E^{-1}(M))$).*

Proof. First it is clear that $M \subseteq E(E^{-1}(M))$. Indeed, if there were $j \in M$ such that $j \notin E(E^{-1}(M))$, we would have $j < j_1$, otherwise $j \in E(E^{-1}(M))$. But $j < j_1$ contradicts the definition of $E^{-1}(M)$.

Up-Set($E(E^{-1}(M))$) gives:

$$\sum_{j\in E(E^{-1}(M))\backslash M} x^j + \sum_{j\in M} x^j \geq |E(E^{-1}(M))\backslash M| + |M| - |E^{-1}(M)| + 1.$$

Summed up to the trivial inequality

$$|E(E^{-1}(M))\backslash M| \geq \sum_{j \in E(E^{-1}(M))\backslash M} x^j$$

we directly obtain $\sum_{j \in M} x^j \geq \overline{\alpha}(M)$. ■

Theorem 3.8. *For any $M \subset \mathcal{N}$, the static up-set inequality (3.2) is facet defining for $\mathscr{P}^n$ if and only if M is a strong up-set extension.*

Proof. ($\Rightarrow$) Let suppose M is not a strong up-set extension, i.e. M is not an up-set, or $E^{-1}(M)$ is not a strong up-set, or does not satisfy the condition $E(E^{-1}(M)) = M$, or does not satisfy condition (iii).

If M is not an up-set then the corresponding static up-set inequality is clearly not a facet.

Let suppose $E^{-1}(M)$ is not an up-set. Then $\overline{E^{-1}(M)}$ is sufficient to cover the demand, and $|M \cap \overline{E^{-1}(M)}| = \overline{\alpha}(M) - 1$ so $\sum_{j \in M} x^j \geq \overline{\alpha}(M)$ is not valid.

Let now suppose $E^{-1}(M)$ is an up-set, but is not minimal. There exists a unit i such that $E^{-1}(M)\backslash\{i\}$ is still an up-set. Then, either $E(E^{-1}(M)\backslash\{i\}) = E(E^{-1}(M))$ or $E(E^{-1}(M)\backslash\{i\}) = E(E^{-1}(M))\backslash\{i\}$. In both cases, Up-Set($E(E^{-1}(M)\{i\})$) dominates Up-Set($E(E^{-1}(M))$). By Lemma 3.9, we can conclude that (3.1) is not a facet of $\mathscr{P}^n$.

Let suppose now that $E^{-1}(M)$ is not strong. By Proposition 3.7, $R = E^{-1}(M)\backslash\{j_1\} \cup \{i_1\}$ is an up-set. As $i_1 \leq j_1$, $E(E^{-1}(M)) \cup \{i_1\} \subset E(R)$. It can be easily checked that Up-Set($E(R)$) dominates Up-Set($E(E^{-1}(M))$). By Lemma 3.9, we can conclude that (3.1) is not a facet of $\mathscr{P}^n$.

Let now suppose that $M \neq E(E^{-1}(M))$. By Lemma 3.9, we can conclude that (3.1) is not a facet of $\mathscr{P}^n$.

Let now suppose $E^{-1}(M)$ does not satisfy condition (iii), which is to say $T = E^{-1}(M)\backslash\{j_1, j_2\} \cup \{1\}$ is an up-set. If $j_1 = 1$ then it means that $T = E^{-1}(M)\backslash\{j_2\}$ is an up-set so $E^{-1}(M)$ is not minimal. We have seen that in this case, (3.1) cannot be a facet of $\mathscr{P}^n$. Otherwise, $j_1 \neq 1$. Then we have:

$$\overline{T} = (\mathcal{N}\backslash M) \cup \{2, 3, ..., \overline{\alpha}(M) - 1, j_1, j_2\}.$$

We consider a vector $\overline{x}$ satisfying (3.1) to equality: $\sum_{j \in M} \overline{x}^j = \overline{\alpha}(M)$. We show that if $\overline{x}^1 = 0$ then $\overline{x}$ is not feasible. Indeed, if the $\overline{\alpha}(M)$ units of M which are up in $\overline{x}$ are the $\overline{\alpha}(M)$ most powerful units of $M\backslash\{1\}$, i.e. $\{2, 3, ..., \overline{\alpha}(M) - 1, j_1, j_2\}$, then, even if all units in $\mathcal{N}\backslash M$ are up, the demand cannot be met, as T is an up-set. So each vector $x \in P$ satisfying $\sum_{j \in M} x^j = \overline{\alpha}(M)$ is such that $x^1 = 1$, so there are less than n linearly independant vertices satisfying $\sum_{j \in M} x^j = \overline{\alpha}(M)$. Thus, (3.1) is not a facet of $\mathscr{P}^n$.

($\Leftarrow$) Let suppose M is a strong up-set extension. We will prove constructively that the hyperplane defined by (3.1) contains n linearly independant vertices of $\mathscr{P}^n$. Let $c = |E^{-1}(M)|$ and $m = |M|$.

Consider the $n \times n$ matrix:

$$X = \begin{bmatrix} U & B_1 & C_{n-m} \\ U & I_c & U \\ C_{m-c} & B_2 & U \end{bmatrix}$$

where B_1 and B_2 are $(n-m) \times c$ and $(m-c) \times c$ respectively, each of them having identical rows of the form

$$b_1 = (1,0,0,...,0),\ b_2 = (1,1,0,...,0).$$

I_c is the identity matrix of order c, U stands for matrices of ones of appropriate dimension. C_p is the $p \times p$ matrix such that C contains zeros on the diagonal, and ones everywhere else.

Each row of X corresponds to a vertex of P. The first $m-c$ columns of X correspond to the units in $M \backslash E^{-1}(M)$, the following c columns correspond to the units in $E^{-1}(M)$ and the last $n-m$ columns correspond to the units in $\mathcal{N} \backslash M$. Each row satisfies (3.1) to equality: among its first m entries, each row has exactly $\overline{\alpha}(M)$ entries equal to 1.

We now prove that each row is a feasible solution, i.e. it belongs to $\mathscr{P}^n$.

As $E^{-1}(M)$ is strong, $E^{-1}(M) \backslash \{j_1\} \cup \{i_1\}$ is not an up-set, so the units in $\overline{E^{-1}(M)} \cup \{j_1\} \backslash \{i_1\}$ suffice to meet the demand. In particular, for any $i \in \mathcal{N} \backslash M$, the units in $\overline{E^{-1}(M)} \cup \{j_1\} \backslash \{i\}$ suffice to meet the demand. Hence the feasibility of the first $n-m$ rows.

As $E^{-1}(M)$ is minimal, for all $i \in E^{-1}(M)$, $E^{-1}(M) \backslash \{i\}$ is not an up-set, so the units in $\overline{E^{-1}(M)} \cup \{i\}$ suffice to cover the demand. Hence the feasibility of the following c rows.

By condition (iii), $A = E^{-1}(M) \backslash \{j_1, j_2\} \cup \{1\}$ is not an up-set. Thus $\overline{A} = \overline{E^{-1}(M)} \cup \{j_1, j_2\} \backslash \{1\}$ is enough to cover the demand by itself. In particular, for all $i \in M \backslash E^{-1}(M)$, the units in $\overline{E^{-1}(M)} \cup \{j_1, j_2\} \backslash \{i\}$ suffice to meet the demand. Hence the feasibility of the last $m-c$ rows. ■

3.4.2 Interval Up-Set inequalities

Let $C \subset \mathcal{N}$ be a subset of units, with $i \in C$, and let $\mathscr{I} = \{t_0, ..., t_1\} \subset \mathcal{T}$ be a time interval of length less than or equal to L^i, *i.e.* $t_1 - t_0 \leq L^i$. The *interval up-set* inequality is defined as follows:

$$\alpha_{\mathscr{I}}(C) + \sum_{t=t_0+1}^{t_1} u_t^i \leq x_{t_1}^i + \sum_{j \in C \backslash \{i\}} \left(x_{t_0}^j + \sum_{t'=t_0+1}^{t_1} u_t^j \right) \tag{3.3}$$

Example 3.2. *Recall the illustrative instance of the MUCP from Example 1.1. Consider $C = \{1,2\}$ and interval $\mathscr{I} = \{1,2\}$. Recall from Example 3.1 that the maximum rank of C on interval $\mathscr{I}$ is $\alpha_{\mathscr{I}}(C) = 1$. For $i = 1$, $L^i = 2$, and the corresponding interval up-set is*

$$1 + u_2^1 \leq x_2^1 + x_1^2 + u_2^2 \tag{3.4}$$

We now give a technical lemma:

Lemma 3.10. *For all $C \subset \mathcal{N}$, $\mathscr{I} = \{t_0, ..., t_1\} \subset \mathcal{T}$ and $k \in \mathscr{I}$, the following holds:*

$$\sum_{j \in C} x_k^j \leq \sum_{j \in C} \left(x_{t_0}^j + \sum_{t=t_0+1}^{t_1} u_t^j \right)$$

Proof. For all $j \in C$, the sum of inequalities (1.4) $x_t^j - x_{t-1}^j \leq u_t^j$ from t_0+1 to k yields $x_k^j \leq x_{t_0}^j + \sum_{t=t_0+1}^{k} u_t^j$.

Hence, summing over all $j \in C$ and using $u_t^j \geq 0 \ \forall t \in \mathcal{T}$, we obtain the proposed inequality. ■

The following result provides a characterization of validity for the interval up-set inequality. For any interval $\mathcal{I} = \{t_0, ..., t_1\} \subset \mathcal{T}$ and for any $i \in \mathcal{N}$, we define the subdivision $Y_{\mathcal{I}}^i = (y_t, t \geq 1)$ of interval $\mathcal{I}$ as follows:

$$\begin{cases} y_0 & = & t_0 \\ y_{t+1} & = & \arg\max_{t' \in \{y_t+1, ..., \min(y_t+\ell^i, t_1)\}} D_{t'} \qquad \forall t \geq 0 \end{cases}$$

Recall that for each subset of units $C \subset \mathcal{N}$, for each time period $t \in \mathcal{T}$ and for each unit $i \in C$, the static i-rank $\overline{\alpha}_t^i(C)$ is the smallest number of units that must be up in M at time t in order to satisfy the residual demand $D_t - \sum_{j \notin M} P_{max}^j$, given that unit i is down at t.

Theorem 3.9 (Validity characterization)**.** *Let $C \subset \mathcal{N}$, for any $i \in C$, for any interval $\mathcal{I} = \{t_0, ..., t_1\} \subset \mathcal{T}$ such that $t_1 - t_0 \leq L^i$, the interval up-set inequality (3.3) is valid for $P_{x,u}^n$ if and only if:*

$\forall y \in Y_{\mathcal{I}}^i$, if $\overline{\alpha}_y^i(C) < \alpha_{\mathcal{I}}(C)$, then each solution $(x,u) \in P_{x,u}^n$ is such that

$$\begin{cases} x_y^i = 1 \\ \ or \\ \sum_{j \in C \setminus \{i\}} \left(x_{t_0}^j + \sum_{t=t_0+1}^{t_1} u_t^j \right) \geq \alpha_{\mathcal{I}}(C) \end{cases} \tag{3.5}$$

Proof. Suppose there exists $y \in Y_{\mathcal{I}}^i$ such that there is a solution (x,u) not satisfying (3.5), *i.e.*

$$\begin{cases} x_y^i = 0 \\ \ and \\ \sum_{j \in C \setminus \{i\}} \left(x_{t_0}^j + \sum_{t=t_0+1}^{t_1} u_t^j \right) < \alpha_{\mathcal{I}}(C) \end{cases}$$

Then, we have $\sum_{t=t_0+1}^{t_1} u_t^i = x_{t_1}^i$. Indeed, as unit i is down at time y, if $x_{t_1}^i = 1$ then $\sum_{t=t_0+1}^{t_1} u_t^i = 1 = x_{t_1}^i$. Otherwise, if $x_{t_1}^i = 0$ then by the min-up inequality $\sum_{t=t_0+1}^{t_1} u_t^i = 0 = x_{t_1}^i$. Thus the interval up-set inequality turns into $\alpha_{\mathcal{I}}(C) \leq \sum_{j \in C \setminus \{i\}} \left(x_{t_0}^j + \sum_{t=t_0+1}^{t_1} u_t^j \right)$. As $\sum_{j \in C \setminus \{i\}} \left(x_{t_0}^j + \sum_{t=t_0+1}^{t_1} u_t^j \right) < \alpha_{\mathcal{I}}(C)$, the interval up-set inequality is violated by solution (x,u).

Now suppose for all $y \in Y_{\mathcal{I}}^i$ such that $\overline{\alpha}_y^i(C) < \alpha_{\mathcal{I}}(C)$, each solution satisfies (3.5).

Let (x,u) be a solution.

- **Case 1**: there exists $y \in Y_{\mathcal{I}}^i$ such that $\overline{\alpha}_y^i(C) < \alpha_{\mathcal{I}}(C)$ and $x_y^i = 0$.

  In this case we have $\sum_{j \in C \setminus \{i\}} \left(x_{t_0}^j + \sum_{t=t_0+1}^{t_1} u_t^j \right) \geq \alpha_{\mathcal{I}}(C)$ so the interval up-set inequality is satisfied by (x,u).

- **Case 2**: there exists $y \in Y^i_{\mathscr{I}}$ such that $\overline{\alpha}^i_y(C) \geq \alpha_{\mathscr{I}}(C)$ and $x^i_y = 0$. By definition of $\overline{\alpha}^i_y(C)$, we know there are at least $\overline{\alpha}^i_y(C)$ units up in C at time y since unit i is down. Thus by Lemma 3.10, we get:
$$\overline{\alpha}^i_y(C) \leq \sum_{j \in C \setminus \{i\}} x^j_y \leq \sum_{j \in C \setminus \{i\}} \left(x^j_{t_0} + \sum_{t=t_0+1}^{t_1} u^j_t \right)$$
Since $\alpha_{\mathscr{I}}(C) \leq \overline{\alpha}^i_y(C)$ we can conclude that the interval up-set inequality is also valid in this case.

- **Case 3**: unit i is up on the whole interval $\mathscr{I}$. By definition, there are at least $\alpha_{\mathscr{I}}(C)$ units in C up at time t_{max}. Thus there are at least $\alpha_{\mathscr{I}}(C) - 1$ units in $C \setminus \{i\}$ up at time t_{max}. By Lemma 3.10,
$$\alpha_{\mathscr{I}}(C) - 1 \leq \sum_{j \in C \setminus \{i\}} x^j_{t_{max}} \leq \sum_{j \in C \setminus \{i\}} \left(x^j_{t_0} + \sum_{t=t_0+1}^{t_1} u^j_t \right)$$
As unit i is up on $\mathscr{I}$, $x^i_{t_1} = 1$ and $\sum_{t=t_0+1}^{t_1} u^i_t = 0$ so the interval up-set inequality is also valid in this case.

- Note that there are no cases left. Indeed, if $x^i_y = 1$ for all $y \in Y^i_{\mathscr{I}}$ then i is up on the whole interval $\mathscr{I}$. If i shuts down at some time $t \in \mathscr{I}$, it remains down at least for ℓ^i time periods. But the difference between two elements y of $Y^i_{\mathscr{I}}$ is at most ℓ^i, by construction of the subdivision $Y^i_{\mathscr{I}}$.

■

Example 3.3. *Recall interval up-set inequality* (3.4) *from Example 3.2, with* $\mathscr{I} = \{1,2\}$, $C = \{1,2\}$ *and* $i = 1$. *In this case,* $Y^1_{\mathscr{I}} = (y_1, y_2)$, *where* $y_1 = 1$ *and* $y_2 = 2$. *Then, for each* $y \in Y^1_{\mathscr{I}}$, $\overline{\alpha}^1_y(C) = 1 = \alpha_{\mathscr{I}}(C)$. *Therefore inequality* (3.4) *is valid.*

Note that the validity condition for the whole polytope from Theorem 3.9 may be hard to check. Let us consider the supporting instance restricted to interval $\mathscr{I}$, denoted by $Inst(\mathscr{I})$. Contrary to the general case, where $\alpha_{\mathscr{I}}(C)$ is hard to compute, the computation of the maximum rank over interval $\mathscr{I}$ for instance $Inst(\mathscr{I})$ is easy. Indeed, $\alpha_{\mathscr{I}}(C) = \overline{\alpha}_{\mathscr{I}}(C)$, as the solution such that the $\overline{\alpha}_{\mathscr{I}}(C)$ most powerful units of C are up on $\mathscr{I}$, alongside with all units in $\mathscr{N} \setminus C$, is a solution to $Inst(\mathscr{I})$.

Let us define $P^n_{x,u}(\mathscr{I})$ the polytope associated to $Inst(\mathscr{I})$. If $\alpha_{\mathscr{I}}(C \setminus \{i\}) < \alpha_{\mathscr{I}}(C)$, and if there exists $y_t \in Y^i_{\mathscr{I}}$ such that $\overline{\alpha}^i_{y_t}(C) < \alpha_{\mathscr{I}}(C)$, then we can easily construct a solution $(x,u) \in P^n_{x,u}(\mathscr{I})$ not satisfying inequalities (3.5). It suffices to set unit i down on interval $[y_{t-1}+1, y_{t-1}+\ell^i]$ and up at all other times, and to set the $\overline{\alpha}_{\mathscr{I}}(C) - 1$ most powerful units of $C \setminus \{i\}$ up on $\mathscr{I}$, alongside with all units in $\mathscr{N} \setminus C$. Consequently, the following result holds, thus providing the necessary and sufficient validity condition for the interval up-set inequality in polytope $P^n_{x,u}(\mathscr{I})$.

Theorem 3.10 (Validity characterization in $P^n_{x,u}(\mathscr{I})$)**.** *Let $C \subset \mathscr{N}$, for any $i \in C$, for any interval $\mathscr{I} = \{t_0, ..., t_1\} \subset \mathscr{T}$ such that $t_1 - t_0 \leq L^i$ and $\alpha_{\mathscr{I}}(C \backslash \{i\}) < \alpha_{\mathscr{I}}(C)$, the interval up-set inequality is valid for $P^n_{x,u}(\mathscr{I})$ if and only if $\forall y \in Y^i_{\mathscr{I}}$, $\overline{\alpha}^i_y(C) \geq \alpha_{\mathscr{I}}(C)$.*

We will see in Theorem 3.15 that in the particular case $\alpha_{\mathscr{I}}(C \backslash \{i\}) = \alpha_{\mathscr{I}}(C)$, the interval up-set inequality (3.3) is dominated by inequalities (1.4) and up-set inequalities (3.1).

A question is whether, for $\Delta \in \mathbb{N}$, there exist other valid inequalities of the following form:

$$\Delta + \sum_{t=t_0+1}^{t_1} u^i_t \leq x^i_{t_1} + \sum_{j \in C \backslash \{i\}} \left(x^j_{t_0} + \sum_{t'=t_0+1}^{t_1} u^j_t \right). \tag{3.6}$$

Interestingly it can be shown that if $\Delta > \alpha_{\mathscr{I}}(C)$ then inequality (3.6) is not valid. On the opposite in the case $\Delta \leq \alpha_{\mathscr{I}}(C) - 1$, inequality (3.6) is valid, but is also dominated by up-set inequalities (3.1), min-up inequalities (1.2) and inequalities (1.4). Thus, the only relevant case is $\Delta = \alpha_{\mathscr{I}}(C)$.

Theorem 3.11. *For all $C \subset \mathscr{N}$, $i \in C$, $\mathscr{I} = \{t_0, ..., t_1\}$ such that $t_1 - t_0 \leq L^i$,*

(i) If $\Delta \leq \alpha_{\mathscr{I}}(C) - 1$, then inequality $\Delta + \sum_{t=t_0+1}^{t_1} u^i_t \leq x^i_{t_1} + \sum_{j \in C \backslash \{i\}} \left(x^j_{t_0} + \sum_{t'=t_0+1}^{t_1} u^j_t \right)$ is valid for $P^n_{x,u}$,

(ii) If $\Delta > \alpha_{\mathscr{I}}(C)$, then inequality $\Delta + \sum_{t=t_0+1}^{t_1} u^i_t \leq x^i_{t_1} + \sum_{j \in C \backslash \{i\}} \left(x^j_{t_0} + \sum_{t'=t_0+1}^{t_1} u^j_t \right)$ is not valid for $P^n_{x,u}$.

Proof. Recall t_{max} denotes the time period at which the demand is maximum on $\mathscr{I}$, and by Lemma 3.5, $\alpha_{t_{max}}(C) = \alpha_{\mathscr{I}}(C)$.

(i): As the length of $\mathscr{I}$ is less than or equal to L^i, from the min-up inequality (1.2) we have

$$\sum_{t=t_0+1}^{t_1} u^i_t \leq \sum_{t=t_1-L^i+1}^{t_1} u^i_t \leq x^i_{t_1}.$$

The up-set inequality for $C \backslash \{i\}$ at time t_{max}, alongside with Lemma 3.10 applied to $C \backslash \{i\}$ and $k = t_{max}$, yields:

$$\alpha_{\mathscr{I}}(C) - 1 = \alpha_{t_{max}}(C) - 1 \leq \alpha_{t_{max}}(C \backslash \{i\}) \leq \sum_{j \in C \backslash \{i\}} x^j_{t_{max}} \leq \sum_{j \in C \backslash \{i\}} \left(x^j_{t_0} + \sum_{t=t_0+1}^{t_1} u^j_t \right)$$

Thus, summing up these two inequalities, we directly obtain the desired inequality with $\Delta \leq \alpha_{\mathscr{I}}(C) - 1$.

(ii): By definition of the rank of C at time t_{max}, note that there exists a solution $(x, u) \in P^n_{x,u}$ such that exactly $\alpha_{t_{max}}(C)$ units in C are up at time t_{max}. Let C_{max} be the set of units in C which are up at time t_{max} in solution (x, u). As in the proof of Lemma 3.5, a solution $(\tilde{x}, \tilde{u})$ can be constructed from (x, u): solution $(\tilde{x}, \tilde{u})$ is such that any unit in $C \backslash C_{max}$ is down on the whole interval $\mathscr{I}$ and any unit in C_{max} is up on the whole interval $\mathscr{I}$.

Thus, there exists a solution $(\tilde{x}, \tilde{u})$ such that any unit in $C \setminus C_{max}$ is down on the whole interval $\mathscr{I}$ and any unit in C_{max} is up on the whole interval $\mathscr{I}$. So the following holds:

$$\alpha_{\mathscr{I}}(C) + \sum_{t=t_0+1}^{t_1} \tilde{u}_t^i = \tilde{x}_{t_1}^i + \sum_{j \in C \setminus \{i\}} \left(\tilde{x}_{t_0}^j + \sum_{t=t_0+1}^{t_1} \tilde{u}_t^j \right).$$

So if $\Delta > \alpha_{\mathscr{I}}(C)$, the inequality is violated by $(\tilde{x}, \tilde{u})$. ■

3.4.3 Generalized interval up-set inequalities

This section introduces valid inequalities, which are generalizations of the interval up-set inequality (3.3). These inequalities will become useful in Section 3.5 to obtain necessary facet conditions for the interval up-set inequality.

Theorem 3.12. *If conditions (3.5) hold, then for each $y \in Y^i_{\mathscr{I}}$ the following inequality is valid:*

$$\alpha_{\mathscr{I}}(C) + (\alpha^i_y(C) - \alpha_{\mathscr{I}}(C))(1 - x^i_y) + \sum_{t=t_0+1}^{t_1} u_t^i \leq x_{t_1}^i + \sum_{j \in C \setminus \{i\}} \left(x_{t_0}^j + \sum_{t=t_0+1}^{t_1} u_t^j \right). \tag{3.7}$$

Proof. If $x^i_y = 1$ then inequality (3.7) translates into the interval up-set inequality, so it is valid as conditions (3.5) hold. If $x^i_y = 0$, *i.e.* unit i is down at time y, there are at least $\alpha^i_y(C)$ units up in $C \setminus \{i\}$ at time y, *i.e.* $\alpha^i_y(C) \leq \sum_{j \in C \setminus \{i\}} x^j_y$. Using Lemma 3.10, we get:

$$\alpha^i_y(C) \leq \sum_{j \in C \setminus \{i\}} x^j_y \leq \sum_{j \in C \setminus \{i\}} \left(x_{t_0}^j + \sum_{t=t_0+1}^{t_1} u_t^j \right).$$

It follows that inequality (3.7) is valid. ■

Note that if for a given $y \in Y^i_{\mathscr{I}}$ $\alpha^i_y(C) = \alpha_{\mathscr{I}}(C)$, then corresponding inequality (3.7) is exactly the interval up-set inequality (3.3).

From inequalities (3.7), another family of valid inequalities generalizing interval up-set inequalities can be derived.

Theorem 3.13. *Let $\beta^i_y = \max_{t \in [\max(y - \ell^i, t_0), y-1]} (\alpha^i_t(C) - \alpha_{\mathscr{I}}(C))$ for all $y \in Y^i_{\mathscr{I}}$. If conditions (3.5) hold, then the following inequality is valid:*

$$\alpha_{\mathscr{I}}(C) + (\alpha^i_{t_1} - \alpha_{\mathscr{I}}(C))(1 - x^i_{t_1}) + \sum_{t=t_0+1}^{t_1} (1 + \beta^i_t) u_t^i \leq x_{t_1}^i + \sum_{j \in C \setminus \{i\}} \left(x_{t_0}^j + \sum_{t=t_0+1}^{t_1} u_t^j \right) \tag{3.8}$$

Proof. The proof can be derived similarly as for Theorem 3.12. ■

Note that if $\beta^i_y \leq 0$ for all $y \in Y^i_{\mathscr{I}}$, then inequality (3.8) is exactly the interval up-set inequality (3.3).

For any $j \in C \backslash \{i\}$, let $C^{i,j}_{\alpha-1}$ be the set of the $\alpha_{\mathscr{I}}(C)-1$ most powerful units of $C \backslash \{i,j\}$. The following theorem shows that another valid inequality exists if conditions (3.5) hold and if there exists $j \in C \backslash \{i\}$ such that unit i, units in $C^{i,j}_{\alpha-1}$ and units in $\mathscr{N} \backslash C$ are not sufficient to cover the demand at time t_1.

Theorem 3.14. *If conditions (3.5) hold and if there exists $j \in C \backslash \{i\}$ such that:*

$$P^i_{max} + \sum_{k \in C^{i,j}_{\alpha-1}} P^k_{max} + \sum_{k \in \mathscr{N} \backslash C} P^k_{max} < D_{t_1}, \tag{3.9}$$

the following inequality is valid:

$$\alpha_{\mathscr{I}}(C) + \sum_{t=t_0+1}^{t_1} u^i_t \leq x^i_{t_1} + x^j_{t_1} + \sum_{k \in C \backslash \{i,j\}} \left(x^k_{t_0} + \sum_{t=t_0+1}^{t_1} u^k_t \right). \tag{3.10}$$

Proof. Consider solution $(x,u) \in P^n_{x,u}$.

- **Case 1**: $x^j_{t_1} = 1$

  Inequality (3.10) translates into

  $$\alpha_{\mathscr{I}}(C) - 1 + \sum_{t=t_0+1}^{t_1} u^i_t \leq x^i_{t_1} + \sum_{k \in C \backslash \{i,j\}} \left(x^k_{t_0} + \sum_{t=t_0+1}^{t_1} u^k_t \right). \tag{3.11}$$

  Unit j is up at time t_1 in solution (x,u) but may be down at another time of $\mathscr{I}$. Thus we define solution $(\overline{x},\overline{u})$ as equal to solution (x,u), except unit j is up at all times in solution $(\overline{x},\overline{u})$. Solution $(\overline{x},\overline{u})$ remains feasible, and since (3.11) does not depend on j, if (x,u) violates inequality (3.11), so does $(\overline{x},\overline{u})$.

  However j is up at all times in $(\overline{x},\overline{u})$ so $\overline{x}^j_{t_0} + \sum_{t=t_0+1}^{t_1} \overline{u}^j_t = 1$. It follows:

  $$\alpha_{\mathscr{I}}(C) + \sum_{t=t_0+1}^{t_1} \overline{u}^i_t > \overline{x}^i_{t_1} + \sum_{k \in C \backslash \{i\}} \left(\overline{x}^k_{t_0} + \sum_{t=t_0+1}^{t_1} \overline{u}^k_t \right)$$

  which is a contradiction, as the interval up-set inequality was supposed to be valid.

- **Case 2**: $x^j_{t_1} = 0$

  As unit j is down at time t_1, there are at least $\alpha_{\mathscr{I}}(C)$ units up in $C \backslash \{i,j\}$ since we assumed that the $\alpha_{\mathscr{I}}(C)-1$ most powerful units of $C \backslash \{i,j\}$ are not sufficient to cover the demand at t_1, even if unit i and units in $\mathscr{N} \backslash C$ are up. So $\alpha_{\mathscr{I}}(C) \leq \sum_{k \in C \backslash \{i,j\}} \left(\overline{x}^k_{t_0} + \sum_{t=t_0+1}^{t_1} \overline{u}^k_t \right)$. With the min-up inequality (1) we can conclude that (3.10) is valid.

■

Note that inequality (3.10) will dominate interval up-set inequality (3.3) under very particular conditions, while inequalities (3.7) and (3.8) present two large classes of inequalities containing interval up-set inequalities.

As a perspective, it seems that other generalizations of interval up-set inequalities could lead to other facet defining inequalities. In particular, it is possible to replace unit $i \in C$ in interval up-set inequalities by a whole subset S of C which plays a role similar to i, as in inequalities of the form

$$\Gamma + \sum_{j \in S} \sum_{t=t_0+1}^{t_1} u_t^j \leq \sum_{j \in S} x_{t_1}^j + \sum_{j \in C \setminus S} \left(x_{t_0}^j + \sum_{t=t_0+1}^{t_1-1} u_t^j \right)$$

and of the form

$$\Gamma' + \sum_{j \in S} \sum_{t=t_0+1}^{t_1} u_{t_1}^j \leq \sum_{j \in S} x_{t_1}^j + \sum_{j \in C \setminus S} \left(x_{t_0}^j + \sum_{t=t_0+1}^{t_1-1} u_t^j \right) + \left(x_{t_0+1}^j + \sum_{t=t_0+2}^{t_1} u_t^j \right)$$

where Γ and Γ' are constants.

3.5 Facial study for interval up-set inequalities

We now explore the cases in which interval up-set inequalities are facet defining for $P_{x,u}^n$. In the following, for given $C \subset \mathcal{N}$, $i \in C$ and $\mathcal{I} = \{t_0, ..., t_1\} \subset \mathcal{T}$ such that $t_1 - t_0 \leq L^i$ and validity conditions from Theorem 3.9 are satisfied, we denote by F the face defined by the interval up-set inequality (3.3):

$$F = \left\{ (x,u) \in P_{x,u}^n \mid \alpha_{\mathcal{I}}(C) + \sum_{t=t_0+1}^{t_1} u_t^i = x_{t_1}^i + \sum_{j \in C \setminus \{i\}} \left(x_{t_0}^j + \sum_{t=t_0+1}^{t_1} u_t^j \right) \right\}$$

3.5.1 Necessary facet conditions in $P_{x,u}^n$

The following theorem presents necessary conditions for the interval up-set inequality to define a facet. Recall that for any $j \in C \setminus \{i\}$, $C_{\alpha-1}^{i,j}$ is the set of the $\alpha_{\mathcal{I}}(C) - 1$ most powerful units of $C \setminus \{i,j\}$.

Theorem 3.15. *If the interval up-set inequality (3.3) defines a facet of $P_{x,u}^n$, then the following conditions hold:*

- $\forall k \in \mathcal{N} \setminus \{i\},\ \forall t \in \mathcal{T} \setminus \{1\},$
$$\begin{cases} \exists (x,u) \in F \text{ such that } u_t^k = 1 & \text{(3.12a)} \\ \exists (x,u) \in F \text{ such that } x_{t-1}^k - x_t^k = 1 & \text{(3.12b)} \\ \exists (x,u) \in F \text{ such that } x_{t'}^k = 0,\ \forall t' \in [t-1, t+\ell^k - 1] & \text{(3.12c)} \end{cases}$$

- $\forall y \in Y_{\mathcal{I}}^i,\ \alpha_y^i(C) \leq \alpha_{\mathcal{I}}(C)$ (3.13)
- $\forall j \in C \setminus \{i\},\ P_{max}^i + \sum_{k \in C_{\alpha-1}^{i,j}} P_{max}^k + \sum_{k \in \mathcal{N} \setminus C} P_{max}^k \geq D_{t_1}$ (3.14)
- $\alpha_{\mathcal{I}}(C \setminus \{i\}) < \alpha_{\mathcal{I}}(C)$ (3.15)

Proof. Conditions (3.12a) – (3.12c) are trivially necessary facet conditions.

Note that if F is a facet of $P^n_{x,u}$, conditions (3.5) hold as the interval up-set is valid.

If (3.13) does not hold, then by Theorem 3.12, inequality (3.7) is valid and dominates the interval up-set inequality.

If (3.14) does not hold, then by Theorem 3.14, inequality (3.10) is valid. Recall that summing up inequalities (1.4) $x^j_t - x^j_{t-1} \leq u^j_t$ from t_0+1 to t_1 yields $x^j_{t_1} \leq x^j_{t_0} + \sum_{t=t_0+1}^{t_1} u^j_t$. It follows that inequality (3.10) dominates the interval up-set inequality.

If (3.15) does not hold, *i.e.* if $\alpha_{\mathscr{I}}(C\backslash\{i\}) = \alpha_{\mathscr{I}}(C)$ then by summing up inequalities (1.4) and up-set inequalities (3.1) and using Lemma 3.10, we get the following inequality:

$$\alpha_{\mathscr{I}}(C) = \alpha_{\mathscr{I}}(C\backslash\{i\}) \leq \sum_{j\in C\backslash\{i\}} \left(x^j_{t_0} + \sum_{t=t_0+1}^{t_1} u^j_t \right).$$

By summing this inequality with the min-up inequality $\sum_{t=t_0+1}^{t_1} u^i_t \leq x^i_{t_1}$ we get the interval up-set inequality. The interval up-set inequality is then dominated by up-set inequalities (3.1), min-up inequalities (1.2) and inequalities (1.4). ■

If one of the conditions (3.13) – (3.15) does not hold, a valid inequality dominating the interval up-set inequality can be derived. Conditions (3.13) – (3.15) are easy to check, while deciding whether (3.12a) – (3.12c) hold is more difficult. Note that condition (3.12b) (resp. (3.12a)) means that unit k shuts down (resp. starts up) at time t in solution (x,u), and that condition (3.12c) states that for any $k \in \mathscr{N}\backslash\{i\}$ and $\forall t \in \mathscr{T}$, there exists a solution $(x,u) \in F$ satisfying property $\Pi_{k,t}$.

3.5.2 Facet characterization in $P^n_{x,u}(\mathscr{I})$

Theorem 3.15 provides necessary conditions for F to define a facet. We now discuss in which cases these conditions are necessary and sufficient. First we give a technical lemma stating that, in any solution $(x,u) \in F$, each unit $j \in C\backslash\{i\}$ starts up at most once on interval $\mathscr{I}$.

Lemma 3.11. *Let $(x,u) \in F$. For all $j \in C\backslash\{i\}$, $x^j_{t_0} + \sum_{t=t_0+1}^{t_1} u^j_t \leq 1$.*

Proof. Suppose there exists $j_0 \in C\backslash\{i\}$ such that $x^j_{t_0} + \sum_{t=t_0+1}^{t_1} u^j_t \geq 2$. We define solution $(\overline{x},\overline{u})$ to be equal to (x,u), except that unit j_0 is up at all times in $(\overline{x},\overline{u})$. Obviously $(\overline{x},\overline{u}) \in P^n_{x,u}$. However, the following holds:

$$\alpha_{\mathscr{I}}(C) + \sum_{t=t_0+1}^{t_1} u^i_t - x^i_{t_1} - \sum_{j\in C\backslash\{i,j_0\}} \left(x^j_{t_0} + \sum_{t=t_0+1}^{t_1} u^j_t \right) \geq 2. \tag{3.16}$$

As $(\overline{x},\overline{u})$ is equal to (x,u) except on (x^j,u^j) coordinates, we can replace $(\overline{x},\overline{u})$ by (x,u) in inequality (3.16). Moreover, as j_0 is up at all times in $(\overline{x},\overline{u})$, we have $x^j_{t_0} + \sum_{t=t_0+1}^{t_1} u^j_t = 1$. Adding this equality to inequality (3.16) we get:

$$\alpha_{\mathscr{I}}(C) + \sum_{t=t_0+1}^{t_1} u_t^i > x_{t_1}^i + \sum_{j \in C \setminus \{i\}} \left(x_{t_0}^j + \sum_{t=t_0+1}^{t_1} u_t^j \right)$$

which means $(\overline{x}, \overline{u})$ violates the interval up-set inequality. As it was assumed to be valid for $P_{x,u}^n$, we have here a contradiction. ■

Now recall $P_{x,u}^n(\mathscr{I})$, the polytope associated to the supporting instance $Inst(\mathscr{I})$ of interval $\mathscr{I}$ and $F_{\mathscr{I}}$ the face of $P_{x,u}^n(\mathscr{I})$ associated to the interval up-set inequality.

Theorem 3.16. *The interval up-set inequality (3.3) defines a facet of $P_{x,u}^n(\mathscr{I})$ if and only if conditions (3.11)-(3.15) hold.*

Proof. The direct implication has been proven in Theorem 3.15. We now prove the return implication.

First for any subset $C_{up} \subset C \setminus \{i\}$ and $t, t' \in \mathscr{I}$, let $v(C_{up}, [t, t'])$ be the vector such that units in subset C_{up} are up at all times of $\mathscr{I}$, units in $C \setminus (C_{up} \cup \{i\})$ are down at all times of $\mathscr{I}$, unit i is up on interval $[t, t']$ and down at all other times, and units in $\mathscr{N} \setminus C$ are up at all times.

Now suppose

$$F_{\mathscr{I}} \subset \{(x,u) \in P_{x,u}^n(\mathscr{I}) \mid \sum_{j \in \mathscr{N}} \left(\sum_{t \in \mathscr{I}} a_t^j x_t^j + \sum_{t \in \mathscr{I} \setminus \{t_0\}} b_t^j u_t^j \right) = \gamma \, (\star)\}$$

where $\gamma \in \mathbb{R}$, and $\forall j \in \mathscr{N}$, $\forall t \in \mathscr{I}$, $a_t^j \in \mathbb{R}$, $b_t^j \in \mathbb{R}$.

We claim that $F_{\mathscr{I}} = \{(x,u) \in P_{x,u}^n(\mathscr{I}) \mid \sum_{j \in \mathscr{N}} \left(\sum_{t \in \mathscr{I}} a_t^j x_t^j + \sum_{t \in \mathscr{I} \setminus \{1\}} b_t^j u_t^j \right) = \gamma\}$, which proves that F is a facet of $P_{x,u}^n$.

Let $k \in \mathscr{N} \setminus C$. There are neither x^k nor u^k variables appearing in the interval up-set inequality, and by condition (3.12c), for all $t \in \mathscr{T}$, there exists a solution $(x,u) \in F$ such that $\Pi_{k,t}$ is satisfied. So by Lemma 3.4 (ii), vectors $\Psi_{t,t+\ell^j}^k(x,u)$ and $\Psi_{t+1,t+\ell^j}^k(x,u)$ are solutions of $F_{\mathscr{I}}$. It follows that $a_t^k = 0$, $t \geq 1$. Furthermore, by condition (3.12a), for any $t \geq t_0$ there is a solution $\chi_{k,t}^u(F_{\mathscr{I}}) \in F_{\mathscr{I}}$ such that unit k starts up at time t. We define $\tilde{\chi}_{k,t}^u(F_{\mathscr{I}})$ to be equal to $\chi_{k,t}^u(F_{\mathscr{I}})$ except that unit k is up at all times. As $\tilde{\chi}_{k,t}^u(F_{\mathscr{I}}) \in F_{\mathscr{I}}$, it follows that $b_t^k = 0$, $t \geq t_0$.

By construction of the subdivision $Y_{\mathscr{I}}^i$, we must have $t_{max} \in Y_{\mathscr{I}}^i$. Thus, by condition (3.13), $\overline{\alpha}_{t_{max}}^i(C) \leq \alpha_{\mathscr{I}}(C)$. So, if we denote by C_α^i the set of the $\alpha_{\mathscr{I}}(C)$ most powerful units of $C \setminus \{i\}$, the units in C_α^i are sufficient to satisfy the demand at time t_{max} (provided that units in $\mathscr{N} \setminus C$ are all up), thus, they are sufficient to satisfy the demand at any time $t \in \mathscr{I}$. Therefore vector $v(C_\alpha^i, \varnothing) \in F_{\mathscr{I}}$. Moreover, for any $t < t_1$, $v(C_\alpha^i, [t_0, t]) \in F_{\mathscr{I}}$. It follows that $a_t^i = 0$, $t \in [t_0, t_1 - 1]$. For any $t > t_0$, we also have $v(C_\alpha^i, [t, t_1]) \in F_{\mathscr{I}}$. We thus get $b_t^i = b_{t_1}^i = -a_{t_1}^i$, $t \in [t_0 + 1, t_1]$.

Let $j \in C \setminus \{i\}$. By condition (3.12b), there exists a solution $\chi_{j,t_0+1}^d(F_{\mathscr{I}}) \in F_{\mathscr{I}}$ such that unit j shuts down at time $t_0 + 1$. By Lemma 3.11, if j shuts down at time $t_0 + 1$, it remains down on $[t_0 + 1, t_1]$ (otherwise we would have $x_{t_0}^j + \sum_{t=t_0+1}^{t_1} u_t^j > 1$). Thus we can define $\chi_{j,t}^d(F_{\mathscr{I}}) \in F_{\mathscr{I}}$, $t \in [t_0 + 1, t_1 + 1]$, as equal to $\chi_{j,t_0+1}^d(F)$ except that j shuts down at time t instead of time $t_0 + 1$

(if $t = t_1 + 1$ then j is up at all times). Thus we get $a^j_t = 0$, $t > t_0$. Similarly, by condition (3.12a), there exists a solution $\chi^u_{j,t_1}(F_{\mathscr{I}}) \in F_{\mathscr{I}}$ such that unit j starts up at time t_1. By Lemma 3.11, unit j is down on interval $[t_0, t_1 - 1]$ in solution $\chi^u_{j,t_1}(F)$. So we can define solutions $\chi^u_{j,t}(F_{\mathscr{I}}) \in F_{\mathscr{I}}$, $t \in [t_0 + 1, t_1]$, as equal to $\chi^u_{j,t_1}(F) \in F$ except that j starts up earlier (at time t instead of time t_1). With these vectors we get $a^j_{t_0} = b^j_{t_1} = b^j_t$, $t > t_0 + 1$.

Now equality ($\star$) is proven to be of the form

$$\gamma + a^i \sum_{t=t_0+1}^{t_1} u^i_t = a^i x^i_{t_1} + \sum_{j \in C \setminus \{i\}} a^j \left(x^j_{t_0} + \sum_{t=t_0+1}^{t_1} u^j_t \right).$$

We prove that $a^i = a^j$, $j \in C$.

Let $j \in C^i_\alpha$ and $t \geq t_0 + 1$. By condition (3.12a), there exists a solution in F such that unit j starts up at time t. Thus, in this solution, j is down at time $t - 1$, and there cannot be more than $\alpha_{\mathscr{I}}(C) - 1$ units of $C \setminus \{i, j\}$ up at time $t - 1$ (otherwise the interval up-set inequality would not be satisfied at equality). This means that at time $t - 1$, the demand can be satisfied by units of $C^{i,j}_{\alpha-1}$, unit i and units of $\mathscr{N} \setminus C$. Moreover, if $t = t_1$, the demand at time t_1 can also be satisfied by those units, as condition (3.14) holds. So $v(C^{i,j}_{\alpha-1}, [t_0, t_1]) \in F_{\mathscr{I}}$. Considering vector $v(C^i_\alpha, \varnothing) \in F_{\mathscr{I}}$, we get $a^j = a^i$.

Let $j \in C \setminus (C^i_\alpha \cup \{i\})$. Recall vector $\chi^d_{j,t_1+1}(F_{\mathscr{I}}) \in F_{\mathscr{I}}$, in which unit j is up at all times. In the solution defined by $\chi^d_{j,t_1+1}(F_{\mathscr{I}})$, there are at most $\alpha_{\mathscr{I}}(C) - 1$ units of $C \setminus \{i, j\}$ up on $\mathscr{I}$ (otherwise $\chi^d_{j,t_1+1}(F_{\mathscr{I}}) \notin F_{\mathscr{I}}$). So there exists a unit $k \in C^i_\alpha$ which is down at all times of $\mathscr{I}$ in solution $\chi^d_{j,t_1+1}(F_{\mathscr{I}})$. We define solution $\tilde{\chi}^d_{j,t_1+1}(F_{\mathscr{I}}) \in F$ as equal to $\chi^d_{j,t_1+1}(F_{\mathscr{I}})$ except that unit j is down at all times, and unit k is up at all times. It follows that $a^j = a^k$. Since $a^k = a^i$, this concludes the proof. ■

Theorem 3.16 states that the interval up-set inequality is facet defining for $P^n_{x,u}(\mathscr{I})$, provided that necessary facet conditions (3.12a) – (3.15) hold.

An interesting problem is to extend the result of Theorem 3.16 to the whole polytope $P^n_{x,u}$. The difficulty is induced by some side effects happening at the outer edges of interval $\mathscr{I}$. However, we can provide some insights into how the result of Theorem 3.16 could be extended to $P^n_{x,u}$. Indeed, in most cases, any vector $(x, u) \in F_{\mathscr{I}}$ introduced in the proof of Theorem 3.16 can be extended to a vector of F. By defining C_{down} as the set of units in C which are down over interval $\mathscr{I}$ in solution (x, u), we can extend vector (x, u) to the whole time horizon T by gradually shutting down the units in C_{down} before time t_0, and then gradually start them up after t_1, so that their minimum down time is satisfied and the demand is met. If such a kind of extension is possible for all the vectors introduced, then it proves that the considered interval up-set inequality defines also a facet of $P^n_{x,u}$, provided that additional vectors of F can be found, using conditions (3.12a) – (3.12c), to show that there are no variables x_t or u_t outside $\mathscr{I}$ defining F.

For example, let us consider $T = 4$, with $D = [20, 10, 10, 20]$, and three units such that $P^i_{max} = 10$, $L^i = 1$, $\ell^i = 2$, $i \in \{1, 2, 3\}$. The interval up-set inequality corresponding to $C = \{1, 2, 3\}$,

$i = 1$ and $\mathscr{I} = \{2,3\}$ defines a facet of $P^n_{x,u}(\mathscr{I})$, from Theorem 3.16. This inequality also defines a facet of $P^n_{x,u}$, as the vectors introduced in Theorem 3.16 can be extended to vectors of $P^n_{x,u}(\mathscr{I})$.

However, in some particular cases, there may be no way to satisfy the demand outside interval $\mathscr{I}$ while satisfying the minimum-down times of units in C_{down}. In these cases, it is likely that interval up-set inequalities are dominated by some stronger inequalities taking into account the demand outside $\mathscr{I}$ which is higher than the demand inside $\mathscr{I}$.

CHAPTER 4

BRANCH & CUT FOR THE MUCP

In this chapter, we study the separation of up-set and interval up-set inequalities, in order to come up with a cutting plane generation procedure to be used in a Branch & Cut algorithm.

The results presented in this chapter have been published in [8].

4.1 Separation

As the rank is already NP-hard to compute, we propose to separate static versions of up-set and interval up-set inequalities, where the rank is replaced by the static rank. We prove that these static inequalities are still NP-hard to separate. We devise a heuristic separation procedure for static up-set inequalities, which takes advantage of facet-defining conditions given in Section 3.4.1. We extend this procedure to separate interval up-set inequalities.

4.1.1 Separation of up-set inequalities

We first consider the separation problem of static up-set inequalities for a given set of units $\mathcal{N}$ with maximum power output P^j_{max}, $j \in \mathcal{N}$, a time horizon $\mathcal{T}$, a demand D_t, $t \in \mathcal{T}$, and a fractional solution $(\overline{x}, \overline{u})$: test whether there exists a set $C \subset \mathcal{N}$ and a time period $t \in \mathcal{T}$ such that $\sum_{j \in \overline{C}} \overline{x}^j_t < \overline{\alpha}_t(C)$, and if yes, then exhibit at least one set C and time period t inducing a violated up-set inequality.

The static up-set inequalities where $\overline{\alpha}_t(C) = 1$ correspond, in the context of the 0-1 knapsack problem, to the cover inequalities, which are known to be NP-complete to separate. The general static up-set inequalities correspond to the extended cover inequalities, whose separation problem's complexity is an open question (see [43]). The following theorem states that the separation problem of static up-set inequalities is NP-complete.

Theorem 4.1. *The separation problem of static up-set inequalities is NP-complete.*

Proof. The separation problem of up-set inequalities is obviously in NP. We prove that the knapsack problem reduces to the separation problem of static up-set inequalities.

Consider a variant of the knapsack problem with n objects associated with weights w^i and values $0 < a^i \leq 1$, $i \in \{1,...,n\}$. Let K be the capacity of the knapsack and let $W < \sum_{i\in\{1,...,n\}} a^i$. The question is whether there exists a subset S of objects such that $\sum_{i\in S} a^i > W$ and $\sum_{i\in S} w^i < K$. Note that this variant can easily be shown to be NP-hard by reduction from the classical knapsack problem.

Let us consider the following instance of the separation problem, where $A = \sum_{i\in\{1,...,n\}} a^i$, $\underline{a} = \min_{i\in\{1,...,n\}} a^i$, $\overline{a} = \max_{i\in\{1,...,n\}} a^i$ and $\lambda = \frac{\underline{a}}{\overline{a}(A-W)}$:

$$\begin{cases} \mathcal{N} = \{1,...,n+1\} \\ T = 1 \text{ and } D_1 = K \\ \forall i \in \{1,...,n\},\ P^i_{max} = w^i \text{ and } P^{n+1}_{max} = D_1 \\ \forall i \in \{1,...,n\},\ \overline{x}^i_1 = \lambda a^i \text{ and } \overline{x}^{n+1}_1 = 1 - \underline{a}/\overline{a}. \end{cases}$$

Note that $\overline{x}^i_1 \in [0,1]$ for any i because it can be supposed w.l.o.g. that $W \leq A - \underline{a}$ (otherwise the only possible solution to the knapsack instance would be to include all objects in the knapsack). Any subset $C \in \mathcal{N}$ of this instance has rank $\alpha_1(C)$ at most 1: indeed, if unit $n+1$ is in C then the demand is satisfied with one unit in C (unit $n+1$), and thus the corresponding rank is at most 1. If unit $n+1$ is not in C then the rank of C is zero since no unit in C is needed to cover the demand.

Here, the separation problem of static up-set inequalities is to find a subset C (containing unit $n+1$) such that $\alpha_1(C) = 1$ and $\sum_{j\in C} \overline{x}^j_1 < 1$. This is the same as finding a subset $\overline{C} = \mathcal{N} \setminus C$ such that $\sum_{i\in\overline{C}} P^i_{max} < D_1$ and $\sum_{j\in\overline{C}} \overline{x}^j_1 > \left(\sum_{j\in\mathcal{N}} \overline{x}^j_1\right) - 1$, *i.e.* $\lambda \sum_{j\in\overline{C}} a^j > \lambda A + \overline{x}^{n+1}_1 - 1$, *i.e* $\sum_{j\in\overline{C}} a^j > W$. A solution to this separation problem is a solution to the knapsack instance, where the elements chosen in the knapsack are exactly the elements in $\overline{C}$. Conversely a solution to the above knapsack instance is a solution to this separation problem. ■

The same proof could be done to show that the separation of extended cover inequalities for the knapsack polytope is NP-complete, thus answering the question raised in [43]. Indeed, any instance of the knapsack problem can be transformed into an extended cover separation problem for instances with n objects, such that objects $\{2,...,n\}$ fit in the knapsack. Thus any cover C will contain object 1, and it follows that $E(C) = C$.

We will see in Section 4.2 that, in practice, these inequalities are very effective. Classically, static up-set inequalities, or extended cover inequalities, are generated by a procedure that searches for cover inequalities and lifts them to stronger inequalities ([4]). Note that [43] propose a heuristic for which the search is based on the construction of a cover set.

We propose an alternate separation algorithm for static up-set inequalities, taking advantage of the facet defining conditions we presented in Section 3.4.1.

Separation algorithm for static up-set inequalities Given a fractionnal solution $(\overline{x}, \overline{u})$, for a given time period t, we first sort the units in non-decreasing order of $\frac{\overline{x}_t^j}{P_{max}^j}$ and store them in a list L. We then construct a set C by iteratively appending units of L, until the corresponding up-set inequality is violated. Hence, we first define the set S which contains the $|C| - \overline{\alpha}_t(C) + 1$ less powerful units of C, *i.e.* units with smallest P_{max}. Finally we remove units from S one by one until obtaining a minimal up-set, and then we swap elements in and out of S to obtain a strong set. Finally, the separation procedure returns the extension of S.

4.1.2 Separation of interval up-set inequalities

In our Branch & Cut algorithm, we consider the following *static interval up-set* inequalities:

$$\overline{\alpha}_{\mathscr{J}}(C) + \sum_{t=t_0+1}^{t_1} u_t^i \leq x_{t_1}^i + \sum_{j \in C \setminus \{i\}} \left(x_{t_0}^j + \sum_{t'=t_0+1}^{t_1} u_t^j \right).$$

From Theorem 3.9, if validity condition $y \in Y_{\mathscr{J}}^i$, $\overline{\alpha}_y^i(C) \geq \overline{\alpha}_{\mathscr{J}}(C)$ holds, these inequalities are valid for $P_{x,u}^n$. Note that by Theorem 3.10, these inequalities correspond exactly to every valid interval up-set inequality for restricted polytope $P_{x,u}^n(\mathscr{J})$.

When $T = 1$, the static interval up-set inequalities are exactly the static up-set inequalities. Since from Theorem 4.1 the separation of static up-set inequalities is an NP-hard problem, we have the following result.

Theorem 4.2. *The separation of static interval up-set inequalities is an NP-hard problem.*

We propose the following separation algorithm for static interval up-set inequality, which is an extension of our algorithm to separate up-set inequalities.

Separation algorithm for static interval up-set inequalities Given a fractionnal solution $(\overline{x}, \overline{u})$, a time interval $[t_0, t_1]$ and a unit i such that $L^i \geq t_0 - t_1$, we first sort the units in non-decreasing order of $\frac{\overline{x}_{t_0}^j + \sum_{t=t_0+1}^{t_1} \overline{u}_t^j}{P_{max}^j}$ and store them in a list L. We then construct a set C by iteratively adding units of L in it, until the corresponding interval up-set inequality is violated. In this case, the separation procedure returns the corresponding set C.

4.2 Experimentation

In this section, some computational results relative to formulation (1.2)–(1.4), (1.7) – (1.10) are presented. To evaluate the effectiveness of up-set and interval up-set inequalities, we separate

them throughout a Branch & Cut tree, using Cplex 12.6.1 with default settings. All the experiments were performed using one thread of a PC with a 64 bits Intel(R) Core(TM) i7-2600K processor running at 3.4GHz, and 16 GB of RAM memory. The problems are solved until optimality (defined within 10^{-6} of relative optimality tolerance) or until the time limit of 3600 seconds is reached.

We compare three methods to solve the (x,u)-formulation (1.2)–(1.4), (1.7)–(1.10) of the MUCP:

- Cplex: Cplex used by its C++ API.
- UP: Branch & Cut algorithm using only up-set cuts, separated with the algorithm given in Section 4.1.1. The cut generation is stopped whenever 300 inequalities have been produced.
- UP+IUP: Branch & Cut algorithm using up-set inequalities as described previously, and interval up-set inequalities. Interval up-set inequalities are separated with the algorithm given in Section 4.1.2 only at the root node when both Cplex and UP algorithm produce no more cuts.

For methods UP and UP+IUP, we also use Cplex C++ API. The separation algorithms are included in Cplex by using the so-called UserCut Callbacks. Note that such a callback deactivates some Cplex features designed to improve the efficiency of the overall algorithm. This may induce a bias when comparing results obtained with and without the use of a UserCut Callback. In order to obtain a non-biased comparison between all the methods, we include in our implementation of Cplex a UserCut Callback which does not separate anything. Note that preliminary results indicate that even if for each MUCP instance considered, the empty UserCut Callback has an impact on the CPU time and the number of nodes, this impact can be positive as well as negative, depending on the instance. Globally, on the MUCP instances considered, there is no significant efficiency loss when using an empty UserCut callback. Therefore, we did not run a comparison with default Cplex, *i.e.* Cplex with no empty callback.

Many parameters are required to define an MUCP instance. Therefore, preliminary experiments were performed to emphasize which parameters affect the performances the most. The time horizon T has low impact on the computation time, as opposed to the number of units n. A fixed time horizon is thus considered for each instance while the number of units n varies depending on the instance class. We set the time horizon to $T = 96$ as it corresponds to the standard value of T in the short term UCP solved at EDF.

The result are presented with respect to instances partitioned into *categories* defined as the triplets (class, symmetries, size): classes R, L, TPR-50, TPR-75 or TPR-100; symmetry type NS or S; and size $n = 10$, $n = 20$ or $n = 50$.

2-peak-demand instances are generated following the procedure given in Section 1.2.4. For each class and size considered, we generate 50 instances with symmetries (S) (with a factor $F = 10$) and 50 instances without symmetries (NS). For each class, we generate instances of various sizes: R and L classes of size $n = 20$ and $n = 50$ and TPR classes of size $n = 10$ and $n = 20$.

Note that for $n = 10$, a symmetry factor $F = 10$ means that there is no symmetry. We therefore only consider (NS) instances when $n = 10$. As all R and L (resp. TPR-100) instances with $n = 20$ are already very well solved (resp. intractable) with Cplex, we generate instances with $n = 50$ (resp. $n = 10$). Note that size $n = 20$ has been considered for all instance classes.

As some instances are already very quickly solved by Cplex, adding new cuts for this kind of instances cannot compensate for the separation time it takes. Thus we want to discriminate between easy instances and hard instances. Then, inside a given instance category, an instance is said to be *hard* if it belongs to the 50% most difficult instances of its category with respect to Cplex computation time, whenever this time exceeds 10 seconds.

The experimental results are presented in two tables as follows.

Table 4.1 displays a comparison between Cplex and UP+IUP for each category of instances. For this purpose, Table 4.1 indicates #H, the number of hard instances, and for each method:

Nodes (N): number of nodes in the Branch & Cut tree.
- Av.: average number
- Min: minimum number
- Max: maximum number

CPU: CPU time (in seconds)
- Av_{all}: average value for all the instances
- Av_H: average value for the hard instances
- Min: minimum value
- Min_H: minimum value for the hard instances
- Max: maximum value

User cuts (only for UP+IUP): number of user cuts, totalizing up-set and interval up-set cuts
- Av.: average number
- Min: minimum number
- Max: maximum number.

Note that the R and L instances of size $n = 20$ were all solved in less than 10 seconds, with an average CPU time less than 1 second. Similarly, the R and L instances of size $n = 50$ without symmetries are easy instances, with an average CPU time of 5 seconds, a maximum CPU time of 41 seconds, and only eight instances with a CPU time exceeding 10 seconds. Another class of instances which can be solved easily is TPR-50, with very few hard instances and a maximum CPU time of 30 seconds.

Table 4.1: Comparative experimental results relative to Cplex and the proposed Branch & Cut algorithms, using UP and IUP inequalities.

			Cplex								**UP+IUP**									
			Nodes (N)			**CPU**					**Nodes (N)**			**User cuts**			**CPU**				
		#H	**Av.**	**Min**	**Max**	**Av**$_{all}$	**Av**$_{H}$	**Min**	**Min**$_{H}$	**Max**	**Av.**	**Min**	**Max**	**Av.**	**Min**	**Max**	**Av**$_{all}$	**Av**$_{H}$	**Min**	**Min**$_{H}$	**Max**
R	(NS)	0	6.78	0	80	0.6386	0	0.29	0	1.75	3.88	0	38	1.84	0	13	0.6598	0	0.3	0	1.83
$n = 20$	(S)	0	16.92	0	318	0.6612	0	0.25	0	1.4	13.52	0	456	2.64	0	23	0.7036	0	0.29	0	1.65
R	(NS)	8	741.48	0	4.48e+03	4.809	14.57	1.11	10.09	24.78	587.18	0	6.68e+03	66.3	0	301	5.808	18	1.11	3.97	41.22
$n = 50$	(S)	25	174451	0	9.9e+05	804.5	1604	1.44	20.4	3600	110953	0	1.02e+06	175	0	466	542.1	1079	1.33	12.61	3600
L	(NS)	0	1.92	0	19	0.6926	0	0.31	0	2.04	1.48	0	15	0.48	0	8	0.743	0	0.33	0	2.18
$n = 20$	(S)	0	20.52	0	161	0.9008	0	0.38	0	2.73	19.02	0	199	2.12	0	14	0.9754	0	0.45	0	2.93
L	(NS)	2	55.88	0	617	2.826	10.09	1.46	10.07	10.12	48.9	0	675	3.02	0	50	3.428	11.29	1.53	10.16	12.41
$n = 50$	(S)	25	340619	0	2.15e+06	888.3	1770	1.85	23.69	3600	211246	0	1.4e+06	56.6	0	301	828.3	1650	1.95	4.06	3600
TPR-50 $n = 10$	(NS)	0	5.66	0	99	0.6296	0	0.16	0	4.26	4.6	0	110	3.96	0	32	0.6316	0	0.15	0	4.28
TPR-50	(NS)	0	34.88	0	414	1.647	0	0.62	0	5.53	33.9	0	331	3.92	0	21	1.717	0	0.62	0	4.79
$n = 20$	(S)	3	339.32	0	3.64e+03	3.569	18.48	0.61	12.43	29.31	388.08	0	4.24e+03	25.6	0	163	4.003	21.61	0.62	14.43	35
TPR-75 $n = 10$	(NS)	13	528.84	0	6.61e+03	11.46	34.2	0.38	11.39	103.4	278.88	0	2.32e+03	82	0	259	8.249	23.16	0.39	8.62	43.38
TPR-75	(NS)	25	2097.36	13	1.51e+04	52.18	89.91	2.74	33.72	332.7	1899.7	19	2.21e+04	186	7	323	53.35	93.14	2.84	25.24	519.42
$n = 20$	(S)	25	28183.1	3	7.11e+05	281.2	541.7	2.06	48.62	3600	27516.4	2	6.54e+05	213	5	334	280.4	538.1	2.73	27.21	3600
TPR-100 $n = 10$	(NS)	25	53653.4	0	2.04e+05	1166	2270	1.19	256.38	3600	47874.3	0	1.8e+05	310	11	463	1188	2297	1.22	171.59	3600
TPR-100	(NS)	25	96232.7	51100	1.34e+05	3600	-	3600	-	3600	92139.6	63800	1.19e+05	311	300	365	3600	-	3600	-	3600
$n = 20$	(S)	25	101096	58885	1.5e+05	3600	-	3600	-	3600	99482.7	50400	1.48e+05	309	0	342	3600	-	3600	-	3600

Table 4.2: Average improvement scores corresponding to comparative experimental results for a selection of instance categories

		UP				UP+IUP			
		I_N	I_{LR}	I_{CPU}	$I_{CPU}(h)$	I_N	I_{LR}	I_{CPU}	$I_{CPU}(h)$
R $n = 50$	**(S)**	62.3%	8.3%	34.1%	44.5%	61.8%	8.4%	19.0%	42.4%
L $n = 50$	**(S)**	42.3%	5.5%	18.9%	25.7%	40.4%	5.6%	9.5%	22.9%
TPR-75 $n = 10$	**(NS)**	25.1%	-0.9%	6.4%	9.3%	44.6%	6.6%	14.9%	30.5%
TPR-75	**(NS)**	16.2%	-0.8%	5.1%	6.7%	27.6%	-1.9%	7.0%	9.9%
$n = 20$	(S)	25.0%	0.2%	8.3%	6.3%	27.7%	2.4%	6.4%	14.6%
TPR-100 $n = 10$	**(NS)**	9.7%	0.07%	-3.7 %	-6.2 %	29%	2.0%	-1.2%	0.6%

When the tightness of the production range increases to 75%, the corresponding instances are much harder than instances with a larger production range (like the instances of classes R and L), but they remain tractable for $n = 20$. The TPR-100 instances appear to be very difficult to solve. For each instance of size $n = 20$, Cplex reaches the time limit of one hour. The instances of size $n = 10$ are already very hard, as the average CPU time is around 1000 seconds. For $n = 20$, and even more for $n = 50$, symmetries deeply affect the computation time.

Table 4.2 provides more details for the comparison of the three methods. As shown in Table 4.1, there is an important variability in the computation time within a given instance category.

We then introduce the improvement score, which is a performance ratio comparing Cplex to one of our methods (UP or UP+IUP) denoted by B&C. The *improvement scores* relative to the number of nodes (N), the CPU time (CPU) and the linear relaxation value at the root (LR) are defined as follows.

$$I_N = 2\frac{N(Cplex)-N(B\&C)}{N(Cplex)+N(B\&C)} \quad I_{CPU} = 2\frac{CPU(Cplex)-CPU(B\&C)}{CPU(Cplex)+CPU(B\&C)} \quad I_{LR} = 2\frac{LR(B\&C)-LR(Cplex)}{LR(Cplex)+LR(B\&C)}$$

For any indicator ind and any two methods m_1 and m_2, the considered improvement score $I_{ind}(m_1, m_2)$ provides a symmetric comparison between the two methods m_1 and m_2. Indeed, the improvement score is a performance ratio, where the reference used is the average between the value from Cplex performance and the value from UP or UP+IUP performance. Using this average value as reference yields the following key property: $I_{ind}(m_1, m_2) = -I_{ind}(m_2, m_1)$. In particular, $I_{ind}(m_1, m_2) \in [-2, 2]$, while the standard relative error calculated as $\frac{ind(m_1)-ind(m_2)}{ind(m_1)} \in [-\infty, 1]$ would be non-symmetric and unbounded.

As we consider a minimization problem, the higher the linear relaxation, the better the lower bound on the optimal solution. Hence for any indicator (N, CPU or LR), the improvement score is positive whenever our method B&C outperforms Cplex with respect to the considered indicator (number of nodes, CPU time or linear relaxation).

Tables 4.2 presents, for both UP and UP+IUP and for a selection of instance categories, the average improvement scores:

I_N: average improvement score relative to the number of nodes in the Branch & Cut tree.

I_{LR}: average improvement score relative to the linear relaxation at the root.

I_{CPU}: average improvement score relative to the CPU time, for all the instances.

$I_{CPU}(h)$: average improvement score relative to the CPU time, for the hard instances.

This table only displays values for the instance categories on which a comparison makes sense, *i.e.* instances which are not easily solved, but still tractable within the time limit.

Observe that both UP and UP+IUP perform very well on the L and R instances.Contrary to the TPR instances where interval up-set inequalities significantly improves the performance of UP, the improvement for L and R instances appears to come from the separation of up-set inequalities. This may seem weird as Cplex and our UP algorithm can produce similar inequalities. This shows that even though the cut generation integrated in Cplex is supposed to be able to produce up-set cuts, our heuristic clearly outperforms Cplex as it finds useful cuts Cplex does not.

Finally note that our methods, UP and UP+IUP, globally outperfom Cplex on the hard instance categories. One objective is to solve TPR-75 and TPR-100 instances, as these instances, close in their structure to ramp-constrained or discrete production units, give us an insight into the potential effectiveness of interval up-set inequalities for the real-world UCP. Interestingly, the TPR-75 instances are solved more efficiently with UP, and even more with UP+IUP. This remark is also true for TPR-100 instances with respect to the relaxation value and the number of nodes, even though it does not show on the CPU time. There may be too many user cuts generated for the hard TPR-100 instances of size $n = 10$: a more dedicated implementation of our Branch&Cut method for this category would be useful.

Table 4.3 compares UP and UP+IUP to Cplex, with respect to the number of instances solved to optimality. For instances on which the time limit is reached, the average optimality gap and best feasible solution value improvement scores are given. The comparison is made on categories where the optimum was not reached on every instance. We do not include (TRP-75, $n = 10$, NS) because there was only one instance over fifty which was not solved to optimality, and all methods have produced the same best feasible solution with similar optimality gaps.
Table 4.3 indicates:

***#solved*:** the number of instances solved to optimality by Cplex

δ**:** difference in terms of the number of instances solved to optimality by UP (resp. UP+IUP) and by Cplex.

I_{best}**:** average “best feasible solution value" improvement score, computed for each instance on which neither Cplex nor UP (resp. UP+IUP) reaches optimality.

Table 4.3: Number of instances solved, average gap improvement score and average best feasible solution improvement score

Instances			Cplex	**UP**			**UP+IUP**		
			#solved	δ	I_{best}	I_{gap}	δ	I_{best}	I_{gap}
R	$n = 50$	(S)	42	2	0%	594%	2	0%	649%
L	$n = 50$	(S)	39	1	0%	251%	1	0%	295%
TPR-75	$n = 20$	(S)	48	0	0%	-29.8%	0	0%	14.7%
TPR-100	$n = 10$	(NS)	38	-1	0.87%	-34.5%	-1	0.983%	13%
TPR-100	$n = 20$	(NS)	0	0	0.0155%	0.468%	0	0.0442%	4.86%
TPR-100	$n = 20$	(S)	0	0	-0.0129%	-1.75%	0	0.00167%	-0.0287%

I_{gap}**:** average gap improvement score, computed for each instance on which neither Cplex nor UP (resp. UP+IUP) reaches optimality.

The optimality gap and best feasible solution value improvement scores are computed the same way the CPU time or the node improvement scores are.

Note that there is not much difference between Cplex and UP (resp. UP+IUP) with respect to the number of instances solved to optimality. Indeed, the very difficult instances are not solved to optimality by any method. Interestingly the best feasible solution values are slightly better with UP+IUP, and to a lesser extent with UP. There is a huge improvement in the optimality gap using UP+IUP, especially on L and R instances.

CONCLUDING REMARKS AND PERSPECTIVES

We propose a polyhedral study of the MUCP with n production units. We first show that the linear relaxation value of the classical formulation ($F^n_{x,u}$) is greater than or equal to that of any demand-coupling formulation. Therefore, we choose to study the polytope defined by ($F^n_{x,u}$), as this formulation involves natural decision variables of the MUCP. We define up-set inequalities, as the translation of the classical extended cover inequalities from the 0-1 knapsack polytope to the MUCP polytope. Interval up-set inequalities are introduced as a generalization of up-set inequalities. This new class of valid inequalities captures both knapsack-like demand constraints and dynamic min-up/min-down constraints, thus are more dedicated to the MUCP.

We completely describe the cases in which these inequalities are valid, and we also characterize the facet defining cases in a restricted polytope. A Branch & Cut algorithm is devised. Compared to Cplex, up-set and interval up-set inequalities used as cuts are particularly efficient for the difficult categories of instances, in particular on the instances where the production range is tighter (TPR75% instances).

From the proof of Lemma 3.10, a part of interval up-set validity is obtained by summation of formulation inequalities. But the proof is not obtained by the Chvátal-Gomory process. An interesting question is then what is the Chvátal-Gomory rank of the interval up-set inequalities.

As pointed out in Section 3.4.3, multiple generalizations of interval up-set inequalities could lead to other facet defining inequalities. More generally, new classes of valid inequalities would be helpful, in particular to solve the TPR-100 instances.

From an experimental point of view, it would be useful to exploit the facet conditions of Theorem 3.15, in order to derive separation algorithms lifting the interval up-set inequalities to the dominating inequality in case condition (3.13) or (3.14) is not satisfied.

Another future work would be to study the ramp-constrained MUCP polytope. It may be particularly useful to lift interval up-set inequalities to the ramp-constrained case, as the ramp-constrained MUCP is close, in its structure, to the TPR75 instances of the MUCP on which interval up-set inequalities are particularly effective.

PART II

BREAKING SYMMETRIES AND SUB-SYMMETRIES

CHAPTER 5

SYMMETRIES AND SUB-SYMMETRIES

5.1 Definitions

We consider an Integer Linear Program (ILP) of the form

$$(ILP) \qquad \min\left\{cx \mid x \in \mathscr{X}\right\}, \text{ with } \mathscr{X} \subset \{0,1\}^n \text{ and } c \in \mathbb{R}^{(m,n)}$$

A symmetry is defined as a permutation π of the indices $\{1,...,n\}$ such that for any solution $x \in \mathscr{X}$, vector $\pi(x)$ is also solution and has same cost, *i.e.*, $\pi(x) \in \mathscr{X}$ and $c(x) = c(\pi(x))$. The *symmetry group* $\mathscr{G}$ of (ILP) is the set of all such permutations.

For instance, consider the problem presented as Example 5.1:

Example 5.1.

$$\min\left\{x_1 + x_2 + 2(x_3 + x_4 + x_5) \ \textit{s. t.}\ 3(x_1 + x_2) + (x_3 + x_4) + 3x_5 = 4 \textit{ and } x \in \{0,1\}^5\right\} \qquad \text{(Ex1)}$$

The symmetry group $\mathscr{G}_1$ of this problem contains $\{id, \pi_{1,2}, \pi_{3,4}\}$, where id is the identity permutation, $\pi_{i,j}$ is the transposition of variables i and j.

Note that subset $S \subset \{1,...,n\}$ and its characteristic vector $x \in \{0,1\}^n$ will be used interchangeably in the following.

The *orbit* of vector $x \in \{0,1\}^n$ under $\mathscr{G}$ is defined as the set of all vectors x' symmetric to x under $\mathscr{G}$:

$$orb(x, \mathscr{G}) = \{x' \in \{0,1\}^n \mid x' = \pi(x),\ \pi \in \mathscr{G}\}$$

Example 5.2. *Referring to Example 5.1, let $x = [1, 1, 1, 0, 0]$. Then $orb(x, \mathscr{G}_1)$ contains $[1, 1, 1, 0, 0]$ and $[1, 1, 0, 1, 0]$, since $\pi_{3,4}(x) = [1, 1, 0, 1, 0]$ and $\pi_{1,2}(x) = x$.*

Vector $y \in \{0,1\}^n$ is said to be *lexicographically greater than* vector $z \in \{0,1\}^n$ if there exists $i \in \{1,...,m-1\}$ such that

- $\forall i' \leq i$, $y_{i'} = z_{i'}$
- $y_{i+1} > z_{i+1}$, *i.e.*, $y_{i+1} = 1$ and $z_{i+1} = 0$.

We write $y \succ z$ (resp. $y \succeq z$) if y is lexicographically greater than z (resp. greater than or equal to z).

Note that y is lexicographically greater than or equal to z if the binary number encoded by y (with the most significant bit on the left) is greater than or equal to the binary number encoded by z, which can be written as:

$$\sum_{i=1}^{n} 2^{n-i} y_i \geq \sum_{i=1}^{n} 2^{n-i} z_i.$$

Vector $y \in orb(S, \mathcal{G})$ is said to be a *representative* among $orb(x, \mathcal{G})$ if y is lexicographically maximum among the vectors in the orbit of x under $\mathcal{G}$, *i.e.*, $y \succeq g(x)$, $\forall g \in \mathcal{G}$.

Example 5.3. *For instance, referring to Example 5.1,* x = [1, 1, 1, 0, 0] *is lexicographically maximal among its orbit, thus x is a representative.*

Classically, (ILP) is solved by Branch & Bound. Whichever branching strategy is chosen, at some point in the branching tree, there will be variables whose values are fixed as a result of the preceding branching decisions taken from the root to the current node. For a given node a of the enumeration tree, F_1^a (resp. F_0^a) is defined as the set of indices of variables fixed to 1 (resp. 0) at node a. F^a is the set of indices of free variables at node a.

Symmetry group at a node a As the branching process fixes variables, the symmetry group $\mathcal{G}(a)$ of the subproblem associated to node a evolves and differs from symmetry group $\mathcal{G}$.

Example 5.4. *Indeed, suppose at a given node a of the enumeration tree relative to problem* (Ex1)*, variable* x_1 *is fixed to 1 and variable* x_2 *is fixed to 0. Then for any feasible solution x at node a,* $\pi_{1,2}(x)$ = [0, 1, x_3, x_4, x_5] *which is not a feasible solution of the subproblem associated to node a. Thus, although* $\pi_{1,2}$ *is in symmetry group* $\mathcal{G}_1$ *of the problem, it is no longer in the symmetry group* $\mathcal{G}(a)$ *associated to a.*

However, as reported in [73], it may be computationally prohibitive to compute the symmetry group for every node of the enumeration tree, since all known algorithms have exponential running time. Thus, an alternative possibility is to consider a subgroup, called $\mathcal{G}^a$ of symmetry group $\mathcal{G}$, defined as follows:

$$\mathcal{G}^a = \{g \in \mathcal{G} \mid g(F_1^a) = F_1^a\}.$$

Example 5.5 proves that at node a, subgroup $\mathcal{G}^a$ of $\mathcal{G}$ may be different from the symmetry group of the subproblem $\mathcal{G}(a)$.

Example 5.5. *Consider problem* $\min\{cx \mid Ax \leq b,\ x \in \{0,1\}^3\}$ *whose solution set is*

$$\{[0, 1, 1], [1, 0, 1], [1, 1, 0], [0, 1, 0]\}.$$

Then $\mathscr{G} = \{id,\ \pi_{1,3}\}$. *Consider a node a of the enumeration such that* $F_1^a = \{3\}$ *and* $F_0^a = \varnothing$. *Then* $\mathscr{G}^a = stab(F_1^a, \mathscr{G}) = id$. *However, the solution set of subproblem a is* {[0, 1, 1], [1, 0, 1]}, *with symmetry group* $\{id,\ \pi_{1,2}\}$.

This example shows how fixed variables in the Branch & Bound enumeration tree can introduce new symmetries in the solution set of the subproblem considered at each node.

Symmetry-breaking techniques If standard Branch & Bound is applied, for each solution x, all elements in $orb(x, \mathscr{G})$ are enumerated in the tree, whereas the optimal value obtained would be the same if only one representative of $orb(x, \mathscr{G})$ were enumerated.

Example 5.6. *Referring to Example 5.1, a Branch & Bound algorithm would produce solutions* $x = [1,1,1,0,0]$ *and* $x' = [1,1,0,1,0]$. *Both are elements of* $orb(x, \mathscr{G}_1)$ *and have value 4. Solution* x' *could have been obtained by applying permutation* $\pi_{3,4}$ *to the representative solution x.*

Various techniques, so called *symmetry-breaking techniques*, are available to handle symmetries in (ILP). A classical idea is, in each orbit of $\mathscr{G}$, to pick one solution, defined as the *representative*, and then restrict the solution set to the set of all representatives.

A technique is said to be *full symmetry-breaking* (resp. *partial symmetry-breaking*) if the solution set is exactly (resp. partially) restricted to the representative set. Moreover, such a technique may introduce some specific branching rules that interfere with the B&B search. This can forbid exploiting a user-defined branching rule or, even, the default branching settings of a state-of-the-art solver. A symmetry-breaking technique is said to be *flexible* if at any node of the B&B tree, the branching rule can be derived from any linear inequality on the variables.

A state-of-the-art of existing symmetry-breaking techniques for arbitrary symmetry group $\mathscr{G}$ is given in Section 5.2. When feasible set $\mathscr{X}$ is a subset of binary matrices of size $m \times n$, and when the symmetry group is the symmetric group $\mathfrak{S}_n$ acting on the columns of solution matrices, *i.e.* the set of all-column permutations, specific symmetry-breaking techniques can be devised. A corresponding state-of-the-art is given in Section 5.3. Applications of symmetry-breaking techniques to the UCP are described in Section 5.4. Finally, in Section 5.5, we introduce sub-symmetries in order to account for symmetries arising from various solution subsets.

5.2 State-of-the-art of generic symmetry-breaking techniques

Many symmetry-breaking techniques rely on the restriction of the feasible set to representatives only. This can be achieved via symmetry-breaking inequalities [27, 53, 54] possibly derived from

the study of the symmetry-breaking polytope [38, 42]. Another option is to use specific branching, pruning or fixing rules during the B&B search [41, 61, 73].

Some symmetry-breaking techniques consist in reformulating the problem so that the new solution set does not feature symmetries. Note that this does not operate a restriction of the feasible set, but propose a symmetry-free reformulation of the problem. This is the case of variable aggregation techniques [50] or decomposition methods such as Branch & Price [6].

5.2.1 Symmetry-breaking inequalities

An inequality $\alpha x \leq \beta$ is said to be a *symmetry-breaking inequality* if for any solution orbit $\mathcal{O}$, there exists at least one element $x \in \mathcal{O}$ which satisfies $\alpha x \leq \beta$. The idea is that many other elements of $\mathcal{O}$ will be cut by inequality $\alpha x \leq \beta$.

A general description of symmetry-breaking inequalities is given in [62], which is a generalization of the framework described in [27]. A closed set $F \subset \mathbb{R}$ is said to be a fundamental region for a symmetry group $\mathcal{G}$ if:

(i) $g(int(F)) \cap int(F) = \varnothing, \quad \forall g \in \mathcal{G}, g \neq id$

(ii) $\cup_{g \in \mathcal{G}} g(F) = \mathbb{R}^n$

where $int(F)$ denotes the interior of F.

If F is a fundamental region for the symmetry group $\mathcal{G}$ of problem (ILP), then the following holds:

$$\min\left\{cx \mid x \in P\right\} = \min\left\{cx \mid x \in P \cap F\right\}$$

Indeed, for any optimal solution x^*, point (ii) guarantees that there exists $g \in \mathcal{G}$ such that $g^{-1}(x^*) \in F$. Point (i) guarantees that region F is not too large.

In [35], a linear description of a fundamental region is proposed:

$$F = \{x \in \mathbb{R}^n \mid (g(\overline{x}) - \overline{x}) \cdot x \leq 0, \forall g \in \mathcal{G}\} \tag{5.1}$$

where $\overline{x} \in \mathbb{R}$ is such that $g(\overline{x}) \neq \overline{x}$ for all $g \in \mathcal{G}$, $g \neq id$.

Thus, any inequality from linear description (5.1) can be added to (ILP). In [62], it is shown that these inequalities can be used even if the condition $g(\overline{x}) \neq \overline{x}$ for all $g \in \mathcal{G}$ does not hold.

Particular case. *Suppose $\mathcal{G}$ contains all permutations of I^n. In order to enforce a lexicographical ordering, one can use inequalities*

$$x_j \geq x_{j+1}, \ \forall j \in \{1, \ldots, n-1\}$$

obtained, for each $j \in \{1, \ldots, n-1\}$, from description (5.1) by setting $\overline{x}_j = n - j$ for all $j \in I^n$ and $g = \pi_{j,j+1}$, the transposition of entries j and $j+1$.

Techniques based on symmetry-breaking inequalities are flexible, since they do not rely on a particular B&B search.

5.2.2 Symmetry-breaking polytopes

For a symmetry group $\mathcal{G}$, the authors of [38] define the symmetry-breaking polytope $P_S(\mathcal{G})$, called *symretope*, as the convex hull of the lexicographically maximal binary points w.r.t. $\mathcal{G}$:

$$P_S(\mathcal{G}) = conv\{x \in \{0,1\}^n \mid x \succeq g(x),\ \forall g \in \mathcal{G}\}$$

The binary points in $P_S(\mathcal{G})$ are exactly those in fundamental region F with $\overline{x}_i = 2^{n-i}$.

As proved in [38], optimization over binary points in symretopes is NP-hard, thus a complete linear description is not available in general. It is still useful to have an IP formulation for binary points in those polytopes in order to handle the symmetries defined by $\mathcal{G}$. Inequalities defined in (5.1) provide such a formulation, but it has exponentially large coefficients and may not be computationally tractable. The authors of [38] consider *symresacks*, a special case of symretopes, where the symmetry group $\mathcal{G}$ contains a unique non-trivial permutation. These polytopes can be seen as knapsack polytopes, where the knapsack constraint has exponential coefficients. This constraint can be replaced by an exponential number of minimal cover inequalities with $\{-1,0,1\}$ coefficients. It is proved in [38] that the separation problem of these minimal cover inequalities for the symresack can be solved in $O(n^2)$ time.

As an arbitrary symretope $P_S(\mathcal{G})$ can be written as the intersection of the symresacks $P_S(g)$ for each $g \in \mathcal{G}$, the authors derive IP formulations with small coefficients for symretopes, with separation in $O(|\mathcal{G}|n^2)$ time.

5.2.3 Pruning by isomorphism in Branch & Bound

A pruning strategy called *isomorphism pruning* (ISP) is defined in [60], such that at any node a, if F_1^a is not a representative then node a is pruned.

The branching strategy called *minimum index branching* (MIB) is defined as branching on the minimum index free variable x_i at each node, with disjunction:

$$x_i = 0 \quad \vee \quad x_i = 1$$

Minimum index branching used alongside with isomorphism pruning can ensure that only representative solutions are explored in the tree, making isomorphism pruning full symmetry-breaking. It can be shown that the optimal value remains the same. For any set $S \subset I^n$ representative under $\mathcal{G}$, subset $S' = S \backslash \{v\}$ with $v = \max\{w \in S\}$ is also a representative. Hence, at a given node a of the Branch & Bound enumeration tree, if F_1^a is not a representative, then any solution S such that $F_1^a \subset S$ is not a representative neither, provided that rule MIB was used in the enumeration tree.

Margot introduces another operation called *0-setting*, which sets to 0 free variables that would induce a non-lexicographically maximum solution.

The 0-setting algorithm consists in the two following operations:

(i) Let b be a node in the enumeration tree and let x_f be the branching variable at b. If a is the son of b where x_f is fixed to 0 then set to 0 all free variables in $orb(\{f\}, \mathscr{G}^a)$.

(ii) Let $f = \min\{r \in F^a\}$. If $F_1^a \cup \{f\}$ is not a representative then set to 0 all free variables in $orb(\{f\}, \mathscr{G}^a)$.

For example, given a variable x_f fixed to 0 by branching at a node a. Suppose some free variable $x_{f'} \in orb(\{f\}, \mathscr{G}^a)$ is fixed to 1 at a descendant node a'. Note that $f' > f$, provided that rule MIB is used. Then for $g \in \mathscr{G}^a$ such that $g(\{f'\}) = \{f\}$, we would have $g(F_1^{a'}) > F_1^{a'}$, thus $F_1^{a'}$ would not be a representative. Thus, variable $x_{f'}$ has value 0 in any representative solution S such that $F_1^a \subset S$.

Margot proves that the use of 0-setting alongside with minimum index branching and isomorphism pruning does not change the optimal value returned by the Branch & Bound.

Example 5.7. *Referring to Example 5.1, consider the enumeration tree of a Branch & Bound algorithm using MIB and ISP. Let node a be such that $F_0^a = \{1\}$ and $F_1^a = \{2\}$. Then F_1^a is not a representative, because $\pi_{1,2}(\{2\}) = \{1\}$ which is lexicographically greater. Thus node a is eliminated by isomorphism pruning.*

Example 5.8. *Referring again to Example 5.1, consider the enumeration tree of a Branch & Bound algorithm using MIB, ISP and 0-setting. Let b be the node such that $F_0^b = \{1\}$ and $F_1^b = \varnothing$. Then $\mathscr{G}^b = \mathscr{G}$ and $orb(\{1\}, \mathscr{G}^b) = \{2\}$. By rule (i) of 0-setting, variable x_2 is set to 0 at node b. Therefore, if 0-setting is used, node a will not be explored by the enumeration tree. It indicates that the use of 0-setting enables early detection and pruning of non-representative solutions in the tree.*

Isomorphism pruning must be used with minimum index branching in order to be valid. Consequently, isomorphism pruning is not flexible.

In practice, the symmetry group $\mathscr{G}$ is represented using the *Schreier-Sims* representation [83]. A backtracking algorithm is proposed to compute $orb(\{f\}, \mathscr{G}^a)$.

In [61], a more flexible branching rule for isomorphism pruning is defined, alongside with more general 0- and 1-setting operations.

5.2.4 Orbital branching

In [73], Ostrowski *et al.* introduce a symmetry-branching strategy called *orbital branching*.

The notion of orbit is extended to variables. We say that $\{x_{i_1}, x_{i_2}, ..., x_{i_k}\}$ is a variable orbit if

$$orb(\{i_1\}, \mathscr{G}) = \{\{i_1\}, \{i_2\}, ..., \{i_k\}\}$$

Let a be a node in the Branch & Bound tree. For a given variable orbit $O = \{x_{i_1}, x_{i_2}, ..., x_{i_k}\}$ of $\mathscr{G}^a$, orbital branching is to branch on the disjunction:

$$x_{i_1} = 1 \quad \vee \quad \sum_{\ell=1}^{k} x_{i_\ell} = 0 \tag{5.2}$$

In [72], the authors extend orbital branching so that the branching disjunction can be based on an arbitrary constraint.

Note that orbital branching is partial symmetry-breaking.

5.2.5 Variable fixing in symmetry-breaking polytopes

At a given node a of the Branch & Bound tree, some variables are set to 0, *i.e.* variables in F_0^a, and some to 1, *i.e.* variables in F_1^a. Based on these variables already fixed by previous branching decisions, some fixings of the remaining free variables can be performed. The idea of *variable fixing* is to restrict the solution space at each node a to be in a given symmetry-breaking polytope P. This is done by fixing to 0 (resp. 1) variables that would yield a solution outside P if fixed to 1 (resp. 0).

This full symmetry-breaking technique is introduced by Kaibel and Pfetsch [41].

Let C^d be the d-dimensional 0/1-cube. A face F of C^d is given by sets $I_0, I_1 \subset I^d$ as follows:

$$F = \{x \in C^d \mid x_i = 0 \ \forall i \in I_0 \text{ and } x_i = 1 \ \forall i \in I_1\}$$

For a polytope $P \subset C^d$ and a face F of C^d defined by (I_0, I_1), the smallest face of C^d that contains $P \cap F \cap \{0,1\}^d$ is denoted by $\text{Fix}_F(P)$, *i.e.* $\text{Fix}_F(P)$ is the intersection of all faces of C^d that contain $P \cap F \cap \{0,1\}^d$.

Example 5.9. *Referring to Example 5.5, consider* $C^3 = \{x \in \mathbb{R}^3,\ 0 \le x_i \le 1$ *for all* $i \in \{1,...,3\}\}$*, and polytope* $P_{ex} \subset C^3$*:*

$$P_{ex} = conv\{[0,\ 1,\ 1],\ [1,\ 0,\ 1],\ [1,\ 1,\ 0],\ [0,\ 1,\ 0]\}.$$

Let F *be the face defined by* $I_0 = \{2\}$ *and* $I_1 = \varnothing$*. Namely,* $F = \{x \in C^3 \mid x_2 = 0\}$*. Then* $P_{ex} \cap F \cap \{0,1\}^3 = [1,\ 0,\ 1]$ *thus* $Fix_F(P)$ *is defined by* $I_0^\star = \{2\}$ *and* $I_1^\star = \{1,3\}$

Lemma 5.1. *If* $Fix_F(P)$ *is a non-empty face, then* $Fix_F(P)$ *is given by sets* $I_0^\star$ *and* $I_1^\star$ *such that*

$$I_0^\star = \left\{(i,j) \mid x_{i,j} = 0 \ \ \forall x \in P \cap F \cap \{0,1\}^{(m,n)}\right\}$$

$$I_1^\star = \left\{(i,j) \mid x_{i,j} = 1 \ \ \forall x \in P \cap F \cap \{0,1\}^{(m,n)}\right\}$$

From an optimization perspective, P can be seen as the polytope defining the solution space $P \cap \{0,1\}^d$. Then, when optimizing over $P \cap \{0,1\}^d$, to each node a of the Branch & Bound tree corresponds a face $F(a)$ defined by F_0^a and F_1^a. The aim is thus to compute sets $I_0^\star$ and $I_1^\star$ defining $\text{Fix}_{F(a)}(P)$, at each node a. Then, if $\text{Fix}_F(P) = \varnothing$ then the node can be pruned. If $\text{Fix}_F(P) \neq \varnothing$, by

Lemma 5.1, any free variable in $I_0^\star$ (resp. $I_1^\star$) can be set to 0 (resp. 1) (otherwise it would yield a solution outside $P \cap F(a) \cap \{0,1\}^d$).

In general, the problem of computing $\mathrm{Fix}_F(P)$ is NP-hard. However, if one can optimize a linear function over $P \cap \{0,1\}^d$ in polynomial time, the fixing $(I_0^\star, I_1^\star)$ at (I_0, I_1) can be computed in polynomial time by solving $2(d - |I_0| - |I_1|)$ many linear optimization problems over $P \cap \{0,1\}^d$ [41].

If sets $I_0^\star$ and $I_1^\star$ relative to P cannot be computed efficiently, some relaxations of P can be considered. Instead of computing $\mathrm{Fix}_F(P)$, one may only compute $\mathrm{Fix}_F(P')$ where $P \subset P'$.

Note that two categories of variable fixing are distinguished in [41]. The process of computing the sets $I_0^\star$ and $I_1^\star$ and fixing the corresponding variables is called *simultaneous* fixing. The second category, called *sequential* fixing, corresponds to the iterated process of searching for additional fixings. Simultaneous fixing is at least as strong as sequential fixing.

5.2.6 Variable aggregation

In [21], Fischetti *et al.* introduce *orbital shrinking*. The idea is to consider only one aggregated variable $z_\omega \in \mathbb{Z}$ per variable orbit ω. Each variable $x_i \in \omega = orb(\{i\}, \mathscr{G})$ is thus replaced by $\frac{z_\omega}{|\omega|} = \frac{1}{|\omega|}\sum_{j\in\omega} x_j$, thus reducing the number of variables. The aggregated program, denoted by (OSR) is a relaxation of the original program (ILP).

For some particular applications, authors [50] have taken advantage of the integer decomposition property (see Theorem 1.3) to prove that the optimal solution of aggregated program (OSR) can be disaggregated into a solution of the original program (ILP).

5.3 State-of-the-art for the symmetric group case

In general, the symmetry group $\mathscr{G}$ acting on the variables is arbitrary. In this section, we consider the *symmetric group case*, *i.e.*, the variable set can be represented as a matrix $x = (x_{i,j})_{i\le m, j\le p}$. and the symmetry group $\mathscr{G}$ is the symmetric group $\mathfrak{S}_n$ acting on the columns of matrix x. This kind of symmetries arise naturally in many scheduling problems.

The ILP considered has the form

$$\min\left\{cx \mid x \in \mathscr{X}\right\}, \text{ with } \mathscr{X} \subseteq \mathscr{P}(m,n) \text{ and } c \in \mathbb{R}^n \tag{5.3}$$

where $\mathscr{P}(m,n)$ is the set of $m \times n$ binary matrices.

Note that in this case, the symmetry group $\mathscr{G}^a$ at a given node a of the Branch & Bound tree can be easily computed: permutations that act on columns j_1, ..., j_k are in $\mathscr{G}(a)$ if and only if for each $i \in \{1,...,m\}$, either variables x_{i,j_1},..., x_{i,j_k} are fixed to the same value or are all free.

For example, this type of symmetry can be found in graph coloring, where symmetry arises in particular from the permutation of colors. If entry $x_{i,j}$ of the solution matrix x corresponds to assigning color j to vertex i, then permuting colors corresponds to permuting columns of matrix x.

5.3.1 Symmetry-breaking inequalities

For the symmetric group case, Margot describes some symmetry-breaking inequalities in [62], obtained from linear description (5.1) for specific values of $\overline{x}$.

The first family of inequalities, usually referred to as Friedman inequalities, enforces a lexicographic order on the columns of x, therefore is full symmetry-breaking:

$$\sum_{i=1}^{m} 2^{m-i} x_{i,j} \geq \sum_{i=1}^{m} 2^{m-i} x_{i,j+1}, \ \forall j \in \{1,...,n-1\}. \tag{5.4}$$

These inequalities state that the binary number encoded by the j^{th} column is greater than or equal to the binary number encoded by the $j+1^{th}$ column.

As the 2^{m-i} term might cause numerical intractability, alternative inequalities featuring binary coefficients can be used, at the expense of losing the full symmetry-breaking property. An option is to use *column inequalities* introduced in [42]:

$$\sum_{k=1}^{i} x_{k,j} \geq x_{i,j+1}, \ \forall j \in \{1,...,n-1\} \tag{5.5}$$

Another option is to use a partial symmetry-breaking form of Friedman inequalities, as in [39, 55, 62]:

$$\sum_{i=1}^{m} x_{i,j} \geq \sum_{i=1}^{m} x_{i,j+1}, \ \forall j \in \{1,...,n-1\} \tag{5.6}$$

The latter inequalities enforce that the total number of ones in each column is non-increasing. Note that this does not enforce lexicographically non increasing columns for the representatives. This ordering is weaker than lexicographic ordering since two different columns can have the same number of ones.

Similarly, the following inequalities, with $\beta^i = i$ or i^2, can be used [62]:

$$\sum_{i=1}^{m} \beta^i x_{i,j} \geq \sum_{i=1}^{m} \beta^i x_{i,j+1}, \ \forall j \in \{1,...,n-1\}.$$

5.3.2 Modified orbital branching

Modified orbital branching, introduced by *Ostrowski et al.* [70], is an extension of orbital branching.

By fixing either one or k variables, orbital branching disjunction often leads to an unbalanced branching tree. In the symmetric group case, it is possible to create a more balanced tree by considering an alternate branching strategy called *modified orbital branching* (MOB).

For a given variable orbit $O = \{x_{i_1}, x_{i_2}, ..., x_{i_k}\}$ of $\mathcal{G}(a)$ at node a, suppose the symmetric group $\mathfrak{S}_{|O|}$ acting on the elements of O is a subset of $\mathcal{G}(a)$. Note that this is always the case when $\mathcal{G}$ is the symmetric group $\mathfrak{S}_n$ acting on the columns of matrix x.

For any $\alpha \in \mathbb{N}$, the following disjunction can be considered:

$$\sum_{\ell=1}^{k} x_{i_\ell} \geq \alpha \quad \vee \quad \sum_{\ell=1}^{k} x_{i_\ell} \leq \alpha - 1$$

For any solution x such that $\sum_{\ell=1}^{k} x_{i_\ell} \geq \alpha$, there exists a permutation $\pi \in \mathfrak{S}_{|O|}$ such that $[\pi(x)]_{i_\ell} = 1$, $\forall \ell \in \{1, ..., \alpha\}$, where $[\pi(x)]_{i_\ell}$ is the $i_\ell{}^{th}$ component of vector $\pi(x)$.

As $\mathfrak{S}_{|O|}$ is a subset of $\mathscr{G}(a)$, $\mathscr{G}(a)$ contains π in particular.

Therefore, the disjunction can be strengthened to:

$$x_{i_\ell} = 1, \ \forall \ell \in \{1, ..., \alpha\} \quad \vee \quad x_{i_\ell} = 0, \ \forall \ell \in \{\alpha, ..., |O|\}$$

Example 5.10. *For example, consider a problem with orbit $O = \{x_1, x_2, x_3\}$. Suppose one uses MOB in the Branch & Bound tree and branches on orbit O at the root node. Then, for $\alpha = 2$, the left child is created by fixing $x_1 = x_2 = 1$ and the right child is created by fixing $x_2 = x_3 = 0$.*

A natural choice for α is $\alpha = \lceil \sum_{k'=1}^{k} \overline{x}_{i_{k'}} \rceil$, where $\overline{x}$ is the solution to the linear relaxation at node a.

Even though modified orbital branching removes a significant proportion of symmetries, it is only partial symmetry-breaking.

In the symmetric group case, Ostrowski *et al.* [70] show the Branch & Bound search with modified orbital branching can be restricted to only representative solutions, making MOB full symmetry-breaking, at the expense of losing the flexibility property. The key idea is to enforce an additional branching rule restricting the variable orbits which can be branched on at each node. Namely, they define the minimum row-index (MI) branching rule which states that variable $x_{i,j}$ is eligible for branching if and only if for all rows $i' < i$, variables $x(i', j)$ have already been fixed. They prove that modified orbital branching alongside with MI branching rule is sufficient to ensure the full symmetry-breaking property. As the MI rule may seem highly restrictive, they also propose some relaxations for which the same property holds, the most flexible branching rule being what is called *relaxed minimum-rank index* (RMRI).

5.3.3 Orbitopes and orbitopal fixing

The convex hull of all $m \times n$ binary matrices with lexicographically non-increasing columns is called a *full orbitope* and is denoted by $\mathscr{P}_0(m, n)$. Special cases of full orbitopes are *packing*, *partitioning* and *covering orbitopes*, which are restrictions to matrices with at most (resp. exactly, at least) one 1-entry in each row.

The key idea is to restrict to $\mathscr{P}_0(m, n)$ the search space of ILP (5.3) in order to explore only lexicographically non-increasing solutions in the Branch & Bound tree.

If matrices in feasible set $\mathscr{X}$ feature exactly (resp. at most, at least) one "1" entry per row, then the search can be restricted to a partitioning (resp. packing, covering) orbitope.

Full orbitopes Full orbitopes can be seen as a special case of symretopes.

The authors of [38] specifically address this particular case, defining *orbisacks* as the convex hull of all $m \times 2$ binary matrices whose first column is lexicographically greater than or equal to the second column. It is shown that only $n-1$ orbisacks need to be considered to obtain an IP-formulation with small coefficients for the full orbitope $\mathscr{P}_0(m,n)$. Furthermore, they prove these inequalities can be separated in $O(mn)$ time.

No complete description of the full orbitope $\mathscr{P}_0(mn)$ is known, and computer experiments conducted in [40] indicate that its facet defining inequalities are extremely complicated. However, exploiting the general framework of polyhedral branching systems defined in [40], a compact $O(mn^3)$ extended formulation is constructed by combining extended formulations of simpler polyhedra. To the best of our knowledge, it has never been used in practice to handle symmetries. For the full orbitope restricted to 2-column matrices, a complete linear description in the x space is available [56].

Packing and partitioning orbitopes In [42], shifted columns inequalities are introduced. The authors prove that these inequalities, together with non-negativity constraints and row-sum inequalities, completely describe both packing and partitioning orbitopes. A polynomial time separation algorithm for the exponentially large class of shifted columns inequalities is also given. Note that these inequalities are full symmetry-breaking.

Orbitopal fixing *Orbitopal fixing* is variable fixing (see Section 5.2.5) when the symmetry-breaking polytope P is an orbitope.

In [41], the authors take advantage of the shifted columns inequalities for partitioning and packing orbitopes in order to characterize the sets $I_0^\star$ and $I_1^\star$ defining $\text{Fix}_F(P)$ where P is the partitioning (or packing) orbitope and F is defined by (F_0^a, F_1^a), at a given node a of the Branch & Bound tree. A linear time orbitopal fixing algorithm is derived for packing and partitioning orbitopes.

It is proved in [41] that for a covering orbitope P, computing the fixing $Fix_F(P)$ is NP-hard.

5.4 State-of-the-art of symmetry-breaking for the UCP

Symmetries in the UCP arise from the existence of groups of identical units, *i.e.*, units with identical characteristics (P_{min}, P_{max}, L, ℓ, c_f, c_0, c_p). The instance is partitioned into *types* $h \in \{1,...,H\}$ of n_h identical units. The unit set of type h is denoted by $\mathscr{N}_h = \{j_1^h,...,j_{n_h}^h\}$.

The solutions of the MUCP can be expressed as a series of binary matrices. For a given type h, we introduce matrix $x^h \in \mathscr{P}(T,n_h)$ such that entry $x_{t,k}^h$ corresponds to variable $x_t^{j_k^h}$, where j_k^h is the index of the k^{th} unit of type h, $k \in \{1,...,n_h\}$. Column j of matrix x^h corresponds to the up/down plan relative to the j^{th} unit of type h. Similarly, we introduce matrices u^h and p^h.

Note that any solution matrix x (resp. u, p) can be partitioned in H matrices x^h (resp. u^h, p^h). Since all units of type h are identical, their production plans can be permuted, provided that the same permutation is applied to matrices x^h, u^h and p^h. Thus, the symmetry group $\mathcal{G}$ contains the symmetric group $\mathfrak{S}_{n_h}$ acting on the columns of x^h, for each unit type h. Consequently, for each type h, feasible solutions x^h can be restricted to be in the $T \times n_h$ full orbitope. As binary variables u are uniquely determined by variables x, breaking the symmetry on x variables will break the symmetry on u variables.

Various types of symmetry-breaking techniques have been applied to the UCP: pruning techniques (such as modified orbital branching), symmetry-breaking inequalities and variable aggregation.

Modified orbital branching for the UCP The authors in [70] apply MOB alongside with several complementary branching rules to break symmetries of the MUCP with ramp and reserve contraints, and down-time dependent start-up costs. Different approaches are compared experimentally: Default Cplex, Callback Cplex, OB (orbital branching), MOB with no branching rules enforced (Cplex is free to choose the next branching variable), and MOB with RMRI (the most flexible branching rule ensuring full symmetry-breaking).

Because advanced Cplex features are turned off when callbacks are used, there is still a huge performance gap between Callback Cplex and Default Cplex. It is shown in [70] that MOB with RMRI is more efficient than MOB, OB and Callback Cplex in terms of CPU time. The difference between using MOB with RMRI and MOB alone is however not as significant as the difference between MOB and simple orbital branching. In particular, referring to the experimental results obtained in [70], the (geometric) average CPU time speed-up between MOB and MOB+RMRI is 1.098.

Symmetry-breaking inequalities As shown in [70], neither Friedman inequalities (5.4) nor column inequalities (5.5) are competitive with respect to the classical UCP formulation when solved by Cplex.

In [55], the partial symmetry-breaking form of Friedman inequalities (5.6) is applied to the UCP. It is shown that on UCP instances similar to those of [70], using these inequalities enables to close the optimality gap much faster than Cplex or Gurobi alone.

Variable aggregation In [50], the authors propose to break symmetries of the UCP by aggregating variables corresponding to identical units. This method is shown to outperform existing symmetry-breaking inequalities.

In the case of the MUCP, variables x, u of formulation (1.2)–(1.8) are aggregated into variables $\widetilde{x}^h_t = \sum_{i \in \mathcal{N}_h} x^i_t \in \{0, ..., n_h\}$ (resp. $\widetilde{u}^h_t = \sum_{i \in \mathcal{N}_h} u^i_t \in \{0, ..., n_h\}$) indicating how many units of type h are up (resp. start up) at time t. Variables $\widetilde{p}^h_t = \sum_{i \in \mathcal{N}_h} p^i_t \in \mathbb{R}$ is the total amount of power produced at

time t by units of type h. Formulation (1.2)–(1.8) becomes

$$\text{A-}(\widetilde{x},\widetilde{u}) \qquad \min_{\widetilde{x},\widetilde{u},\widetilde{p}} \quad \sum_{h=1}^{H}\sum_{t=1}^{T} c_f^h \widetilde{x}_t^h + c_p^j \widetilde{p}_t^h + c_0^j \widetilde{u}_t^h$$

$$\text{s. t.} \quad \sum_{t'=t-L^h+1}^{t} \widetilde{u}_{t'}^h \le \widetilde{x}_t^h \qquad \forall h \in \mathcal{N}_h,\ \forall t \in \{L^h,...,T\} \tag{5.7}$$

$$\sum_{t'=t-\ell^h+1}^{t} \widetilde{u}_{t'}^h \le n_h - \widetilde{x}_{t-\ell^h}^h \qquad \forall h \in \mathcal{N}_h,\ \forall t \in \{\ell^h,...,T\} \tag{5.8}$$

$$\widetilde{u}_t^h \ge \widetilde{x}_t^h - \widetilde{x}_{t-1}^h \qquad \forall h \in \mathcal{N}_h,\ \forall t \in \{2,...,T\} \tag{5.9}$$

$$P_{min}^h \widetilde{x}_t^h \le \widetilde{p}_t^h \le P_{max}^h \widetilde{x}_t^h \qquad \forall h \in \mathcal{N}_h,\ \forall t \in \mathcal{T} \tag{5.10}$$

$$\sum_{h=1}^{H} \widetilde{p}_t^h \ge D_t \qquad \forall t \in \mathcal{T} \tag{5.11}$$

$$\widetilde{x}_t^h, \widetilde{u}_t^h \in \{0,...,n_h\} \qquad \forall h \in \mathcal{N}_h,\ \forall t \in \mathcal{T} \tag{5.12}$$

When aggregating variables corresponding to h identical units, one must ensure that the aggregated production plan, satisfing (5.7)–(5.10), can be disaggregated into h feasible production plans satisfying (1.2)–(1.6). Inequalities (1.2)–(1.6) have the *integer decomposition property* [7], *i.e.*, any integer solution $(\widetilde{x},\widetilde{u},\widetilde{p})$ of formulation (5.7)–(5.12) can be disaggregated into an integer solution (x,u,p) of formulation (1.2)–(1.8). A disaggregation algorithm for the MUCP is proposed in [50].

When ramp constraints are considered in formulation (1.2)–(1.8), the integer decomposition property is lost. Examples of aggregated solutions which cannot be disaggregated are given in [50].

As the integer decomposition property depends on the formulation considered, an interval formulation is introduced in [50] for the ramp-constrained MUCP. It corresponds to the interval formulation presented in Section 1.2.5, where inequalities (1.20) defining the feasible production polytope of each unit i include both production limits (5.13) and ramp constraints (5.14)-(5.17):

$$P_{min}^j \le p_t^i(t_0,t_1) \le P_{max}^i \tag{5.13}$$

$$p_t^i(t_0,t_1) - p_{t-1}^i(t_0,t_1) \le RU^i \qquad \forall t \in \{t_0+1,...,t_1-1\} \tag{5.14}$$

$$p_{t-1}^i(t_0,t_1) - p_t^i(t_0,t_1) \le RD^i \qquad \forall t \in \{t_0+1,...,t_1-1\} \tag{5.15}$$

$$p_{t_0}^i(t_0,t_1) \le SU^i \tag{5.16}$$

$$p_{t_1-1}^i(t_0,t_1) \le SD^j \tag{5.17}$$

Inequalities (1.20)–(1.21) have the integer decomposition property, thus variables $y^i(t_0,t_1)$ (resp. $p_t^i(t_0,t_1)$) can be aggregated into variables $\widetilde{y}^h(t_0,t_1) = \sum_{i\in\mathcal{N}_h} y^i(t_0,t_1)$ and $\widetilde{p}_t^i(t_0,t_1) = \sum_{i\in\mathcal{N}_h} p_t^i(t_0,t_1)$, leading to aggregated formulation Int($\widetilde{y}$):

$$\text{Int}(\widetilde{y}) \qquad \min_{\widetilde{y},\widetilde{p}} \quad \sum_{h=1}^{H} \sum_{\{t_0,\dots,t_1-1\}\in\mathscr{Y}_h} c^h_{t_0,t_1}\widetilde{y}^h(t_0,t_1) + c^h_p \sum_{t=t_0}^{t_1-1} \widetilde{p}^h_t(t_0,t_1)$$

$$\text{s. t.} \qquad A^h(t_0,t_1)\widetilde{p}^h(t_0,t_1) \le b^h(t_0,t_1)\widetilde{y}^h(t_0,t_1) \quad \forall h,\ \forall\{t_0,\dots,t_1-1\}\in\mathscr{Y}_h \tag{5.18}$$

$$\sum_{\substack{\{t_0,\dots,t_1-1\}\in\mathscr{Y}_h \\ \text{s.t. } t\in\{t_0,\dots,t_1+\ell^h\}}} \widetilde{y}^h(t_0,t_1) \le n_h \quad \forall h,\ \forall t\in\mathscr{T} \tag{5.19}$$

$$\sum_{h=1}^{H} \sum_{\{t_0,\dots,t_1-1\}\in\mathscr{Y}_h} \widetilde{p}^h_t(t_0,t_1) \ge D_t \quad \forall t\in\mathscr{T} \tag{5.20}$$

$$\widetilde{y}^h(t_0,t_1)\in\{0,\dots,n_h\} \quad \forall h,\ \forall\{t_0,\dots,t_1-1\}\in\mathscr{Y}_h \tag{5.21}$$

where $\mathscr{Y}_h = \{\{t_0,\dots,t_1-1\}\in T\times T \mid t_1-t_0\ge L^h\}$, and L^h (resp. ℓ^h) is the minimum up time of units of type h.

5.5 Sub-symmetries and sub-orbitopes

As stated in Section 5.1, at a given subproblem a of the branching tree, the symmetry group $\mathscr{G}(a)$ is different from $\mathscr{G}$ and may contain symmetries undetected in $\mathscr{G}$. While some generic symmetry-breaking techniques (as isomorphism pruning or orbital branching) are suited to take into account such symmetries arising at a given node, in practice it is too expensive to recompute the symmetry group at each node of the branching tree. Therefore, only a subgroup $\mathscr{G}^a$ of $\mathscr{G}(a)$ is usually considered (see Section 5.1). Subgroup $\mathscr{G}^a$ contains only the symmetries already detected in the symmetry group $\mathscr{G}$ of the original problem. However for many problems, symmetries of $\mathscr{G}$ can be deduced from the problem's structure, and so can symmetries of $\mathscr{G}(a)$, for some particular subproblems a. In this case, symmetries of $\mathscr{G}(a)$ do not need to be computed during the Branch & Bound procedure, and may be handled together with symmetries of $\mathscr{G}$.

In Section 5.5.1, we generalize symmetries to take into account known symmetries arising in a given set of subproblems. A similar notion has been introduced in the context of Constraint Satisfaction Programming [29, 30]. Symmetries corresponding to such subproblems can be tackled with symmetry-breaking inequalities, or during the B&B search. In Section 5.5.2, we generalize the concept of full orbitope to take into account subproblems featuring symmetric groups as symmetry group.

5.5.1 Sub-symmetries

Consider a subset $Q\subset\mathscr{X}$ of solutions of ILP (5.3). The sub-symmetry group $\mathscr{G}_Q$ relative to subset Q is defined as the symmetry group of subproblem $\min\{cx \mid x\in Q\}$. For instance, such subset

$Q \subset \mathscr{X}$ can correspond to a B&B node, defined as solutions satisfying branching inequalities.

Permutations in sub-symmetry group $\mathscr{G}_Q$ are referred to as *sub-symmetries*. The main motivation to look at sub-symmetries in $\mathscr{G}_Q$ is that they remain undetected in the symmetry group $\mathscr{G}$ of the problem. This is illustrated in the following example.

Example 5.11. *Consider an ILP whose solution set is* $\mathscr{X} = \{X_1, X_2, Y\} \subset \{0,1\}^{(1,3)}$*, where*

$$X_1 = [1, 0, 1], \quad X_2 = [1, 1, 0], \quad Y = [0, 1, 0].$$

Suppose also solutions X_1 *and* X_2 *have same cost, i.e.,* $c(X_1) = c(X_2)$*. Consider solution subset* $Q \subset \mathscr{X}$ *such that* $Q = \{X \in \mathscr{X} \mid X(1,1) + X(1,2) + X(1,3) = 2\}$*, then* $Q = \{X_1, X_2\}$*. Now consider transposition* π_{132} *such that* $\pi_{132}(X) = [X(1,1), X(1,3), X(1,2)]$*. Obviously,* π_{132} *is in sub-symmetry group* $\mathscr{G}_Q$*, but not in symmetry group* $\mathscr{G}$*, as* $\pi_{132}(Y) = [0, 0, 1] \notin \mathscr{X}$*.*

Property 5.1. *Two solutions in the same orbit with respect to a sub-symmetry group* $\mathscr{G}_Q$ *may not be in the same orbit with respect to the symmetry group* $\mathscr{G}$*.*

Referring to Example 5.11, solutions X_1 and X_2 are in the same orbit with respect to $\mathscr{G}_Q$ since $\pi_{132} \in \mathscr{G}_Q$. To see that solutions X_1 and X_2 are not in the same orbit with respect to $\mathscr{G}$, it is sufficient to show that there is no permutation $\pi \in \mathscr{G}$ such that $\pi(X_1) = X_2$. Suppose there was such a permutation π. First note that $\pi_{132} \notin \mathscr{G}$ thus $\pi \neq \pi_{132}$. Since both X_1 and X_2 have exactly one entry to 0, π must be such that $\pi(e_2) = e_3$, where, for $i \in \{1,...,3\}$ $e_i \in \{0,1\}^{(1,3)}$ is such that $e_i(1,i) = 1$ and $e_i(1,j) = 0$, $\forall j \neq i$. Since $Y = e_2$, $\pi(Y) = e_3 \notin \mathscr{X}$, which is a contradiction. Thus, X_1 and X_2 are not in the same orbit with respect to the symmetry group $\mathscr{G}$, which shows the symmetry acting between these two solutions is not detected in $\mathscr{G}$.

We now generalize to sub-symmetries the concepts introduced for symmetries.

Let $\{Q_i \subset \mathscr{X},\ i \in \{1,...,s\}\}$ be a set of matrix subsets. To each Q_i, $i \in \{1,...,s\}$, corresponds a sub-symmetry group $\mathscr{G}_{Q_i}$ containing sub-symmetries that may not be detected in the symmetry group $\mathscr{G}$. Let O^i_k, $k \in \{1,...,o_i\}$, be the orbits defined by $\mathscr{G}_{Q_i}$ on subset Q_i, $i \in \{1,...,s\}$.

When considering only the symmetry group $\mathscr{G}$, the orbits of the solutions form a partition of the solution set $\mathscr{X}$. However, the set $\mathscr{O} = \{O^i_k,\ k \in \{1,...,o_i\},\ i \in \{1,...,s\}\}$ of orbits defined by sub-symmetry groups $\mathscr{G}_{Q_i}$, $i \in \{1,...,s\}$, does not form a partition of $\mathscr{X}$ anymore. Indeed, for given $i, j \in \{1,...,s\}$, if $Q_i \cap Q_j \neq \emptyset$, then any $x \in Q_i \cap Q_j$ will appear in both the orbits of $\mathscr{G}_{Q_i}$ and the orbits of $\mathscr{G}_{Q_j}$. In order to break such sub-symmetries, removing all non-representatives of an orbit of $\mathscr{G}_{Q_i}$ may remove the representative of an orbit of $\mathscr{G}_{Q_j}$, thus leaving the latter unrepresented.

We thus generalize the concept of orbit in order to define a new partition of $\mathscr{X}$ taking sub-symmetries into account. First, for given $X \in \mathscr{P}(m,n)$, let us define $\mathscr{G}(X)$, the set of all permutations π in $\bigcup_{i=1}^{s} \mathscr{G}_{Q_i}$ such that π can be applied to X:

$$\mathscr{G}(X) = \bigcup_{Q_i \ni X} \mathscr{G}_{Q_i}$$

We now define a relation $\mathscr{R}$ over the solution set $\mathscr{X}$. Matrix X' is said to be in relation with X, written $X' \mathscr{R} X$, if

$$\exists r \in \mathbb{N}, \exists \pi_1, ..., \pi_r \mid \pi_k \in \mathscr{G}(\pi_{k-1}...\pi_1(X)), \forall k \in \{1, ..., r\}, \text{ and } X' = \pi_1\pi_2...\pi_r(X).$$

The *generalized orbit* $\mathbb{O}$ *of* X *with respect to* $\{Q_i, i \in \{1, ..., s\}\}$ is thus the set of all X' such that $X' \mathscr{R} X$. Roughly speaking, orbits that intersect one another are collected into generalized orbits. Matrix X' is in the generalized orbit of X if X' can be obtained from X by composing permutations of groups $\mathscr{G}_{Q_i}$, ensuring that each permutation $\pi \in \mathscr{G}_{Q_i}$ is applied to an element of Q_i. Note that $\mathscr{R}$ is an equivalence relation, thus the set of all generalized orbits with respect to $\{Q_i, i \in \{1, ..., s\}\}$ is a partition of $\mathscr{X}$. Moreover, for a given $X \in \mathscr{X}$, each X' in the generalized orbit of X is such that $X' \in \mathscr{X}$ and $c(X') = c(X)$. Note that the generalized orbit may not be an orbit of any of the symmetry groups $\mathscr{G}_{Q_i}$, $i \in \{1, ..., s\}$.

Remark 5.1. *By definition, for any generalized orbit* $\mathbb{O}$*, there exist orbits* σ_1*, ...,* $\sigma_p \in \mathscr{O}$ *such that* $\mathbb{O} = \cup_{i=1}^{p} \sigma_i$.

Note that the union $\mathbb{O} = \cup_{i=1}^{p} \sigma_i$ may contain several orbits relative to the same subset Q_i.

Example 5.12. *Consider an ILP having the following feasible solutions:*

$$X_1 = [1, 1, 0, 0], \quad X_2 = [1, 0, 0, 1], \quad X_3 = [0, 0, 1, 1], \quad X_4 = [0, 1, 1, 0], \quad X_5 = [0, 1, 0, 1]$$

$$Y_1 = [1, 0, 0, 0], \quad Y_2 = [0, 0, 0, 1].$$

with $c(X_1) = c(X_2) = c(X_3) = c(X_4) = c(X_5)$ *and* $c(Y_1) = c(Y_2)$.

Let $Q_1 = \{X_1, X_2, X_3, X_4\}$, $Q_2 = \{X_1, X_5\}$, $Q_3 = \{X_4, X_5\}$ *and* $Q_4 = \{Y_1, Y_2\}$. *The permutation sending* X *to* $[X(1,j_1), X(1,j_2), X(1,j_3), X(1,j_4)]$ *is denoted by* $\pi_{j_1j_2j_3j_4}$. *Note that* $\pi_{2341} \in \mathscr{G}_{Q_1}$, $\pi_{4231} \in \mathscr{G}_{Q_2}$, $\pi_{1243} \in \mathscr{G}_{Q_3}$ *and* $\pi_{4231} \in \mathscr{G}_{Q_4}$. *Thus, the generalized orbit of* X_1 *with respect to* $\{Q_1, Q_2, Q_3, Q_4\}$ *is* $\{X_1, X_2, X_3, X_4, X_5\}$, *as* $X_2 = \pi_{2341}(X_1)$, $X_3 = \pi_{2341}(X_2)$, $X_4 = \pi_{2341}(X_3)$ *and* $X_5 = \pi_{1243}(X_4)$. *Similarly, the generalized orbit of* Y_2 *with respect to* $\{Q_1, Q_2, Q_3, Q_4\}$ *is* $\{Y_1, Y_2\}$. *All in all there are two generalized orbits* $O = \{X_1, X_2, X_3, X_4, X_5\}$ *and* $O' = \{Y_1, Y_2\}$. *Note that* O' *corresponds to the single orbit* Q_4.

While simple orbits $\sigma \in \mathscr{O}$ may sometimes be easily determined, the generalized orbits may anyway be difficult to compute. In this case, one may want to choose a representative $r(\sigma) \in \sigma$ for each orbit $\sigma \in \mathscr{O}$, and then use a *sub-symmetry-breaking* technique to remove all elements $\sigma \backslash r(\sigma)$ from the search, for each orbit $\sigma \in \mathscr{O}$. As for given orbit σ, the set $\sigma \backslash r(\sigma)$ may contain the representative of another orbit σ', we need to ensure that there remains at least one element per generalized orbit after the removal of all elements $\cup_{\sigma \in \mathscr{O}}(\sigma \backslash r(\sigma))$. To this end the choice of the representatives $r(\sigma)$ must satisfy the following compatibility property.

Definition 5.1. The set of representatives $\{r(\sigma), \sigma \in \mathscr{O}\}$ is said to be *orbit-compatible* if for any generalized orbit $\mathbb{O} = \cup_{i=1}^{p} \sigma_i$, $\sigma_1, ..., \sigma_p \in \mathscr{O}$, there exists j such that $r(\sigma_j) = r(\sigma_i)$ for all i such that $r(\sigma_j) \in \sigma_i$. Such a solution $r(\sigma_j)$ is said to be a *generalized representative* of $\mathbb{O}$.

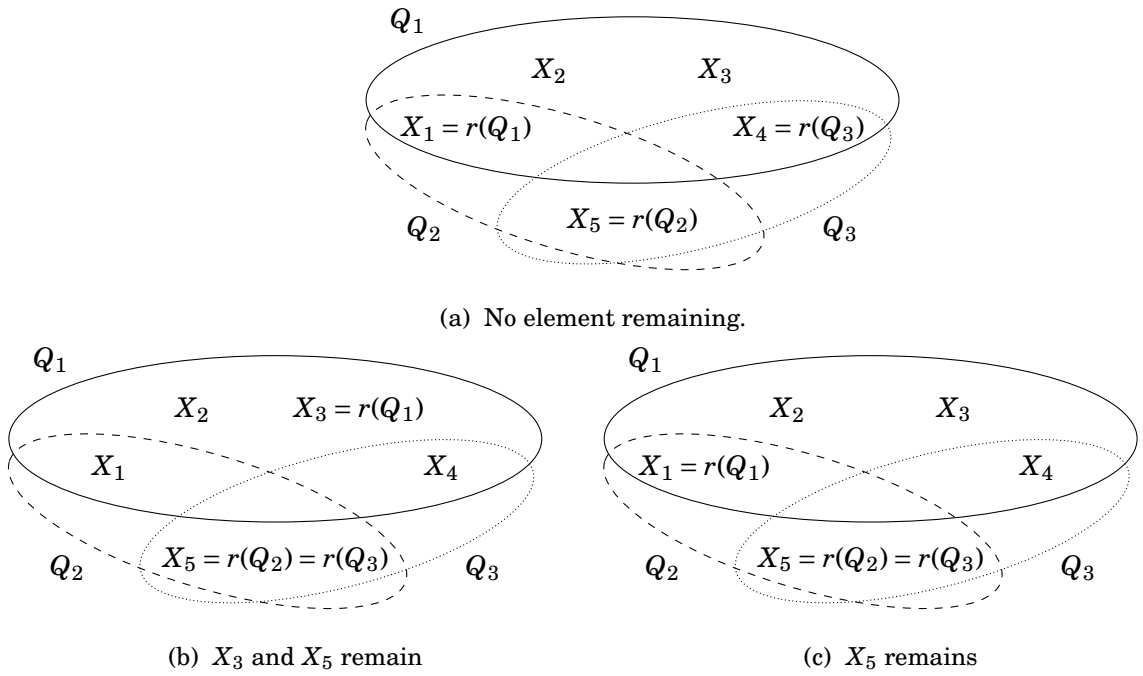
Note that there always exists a set of orbit-compatible representatives: start by choosing a representative $r(\sigma)$ for a given $\sigma \in \mathscr{O}$, and then choose $r(\sigma)$ as the representative of each orbit σ' in which $r(\sigma)$ is contained. Representatives of orbits not containing $r(\sigma)$ can be chosen arbitrarily.

There may exist several generalized representatives of a given generalized orbit. If $\{r(\sigma), \sigma \in \mathscr{O}\}$ is orbit-compatible then for each generalized orbit $\mathbb{O} = \cup_{i=1}^{p} \sigma_i$ there exists $i \in \{1, ..., p\}$ such that either $r(\sigma_i)$ is not contained in any other orbit $\sigma_j \in \mathscr{O}$, $j \neq i$, or $r(\sigma_i)$ is the representative of any orbit to which it belongs. The next lemma states that when representatives are orbit-compatible, there remains at least one element per generalized orbit even if all elements $\cup_{\sigma \in \mathscr{O}} (\sigma \setminus r(\sigma))$ have been removed.

Lemma 5.2. *For given orbit-compatible representatives $r(\sigma)$, $\sigma \in \mathscr{O}$, for any generalized orbit $\mathbb{O} = \cup_{i=1}^{p} \sigma_i$, $\sigma_1, ..., \sigma_p \in \mathscr{O}$, $\exists j \in \{1, ..., p\}$ such that $r(\sigma_j) \notin \cup_{i=1}^{p} (\sigma_i \setminus r(\sigma_i))$.*

Note that even if the set of representatives is orbit-compatible, it may happen that an entire orbit $\sigma \in \mathscr{O}$ is removed by a sub-symmetry-breaking technique. However, if orbit-compatibility is satisfied, there will always remain at least one element in the corresponding generalized orbit, with same cost as any solution in orbit σ.

Referring to Example 5.12, we focus on generalized orbit O. In Figure 5.1a, X_1 (resp. X_4, X_5) is chosen to be the representative of orbit Q_1 (resp. Q_3, Q_2). The set of chosen representatives is not orbit-compatible. Indeed, there is no generalized representative as each representative belongs also to another orbit, of which it is not representative. Thus the set of removed elements $\cup_{i=1}^{p} (\sigma_i \setminus r(\sigma_i))$ contains all elements of the generalized orbit. In Figure 5.1b, X_3 (resp. X_5) is chosen to be representative of Q_1 (resp. orbits Q_3 and Q_2). In this case, the set of chosen representatives is orbit-compatible, since solutions X_3 and X_5 are generalized representatives of O. Indeed, X_3 is representative of Q_1 and does not belong to any other orbit, so it remains after removal removal of $\cup_{i=1}^{p} (\sigma_i \setminus r(\sigma_i))$. Solution X_5 is representative of Q_2 and Q_3, and belongs to these two orbits only, so it remains after removal as well. In Figure 5.1c, X_1 (resp. X_5) is chosen to be representative of Q_1 (resp. orbits Q_3 and Q_2). In this case, the set of chosen representatives is orbit-compatible, since solution X_5 is a generalized representative of O. Indeed, X_5 is representative of Q_2 and Q_3 and does not belong to any other orbit. This choice of representatives is certainly the best as there is exactly one generalized representative of O. Indeed, X_1 is representative of Q_1, but also belongs to orbit Q_2 which has another representative. Therefore, X_1 is in the set of removed elements $\cup_{i=1}^{p} (\sigma_i \setminus r(\sigma_i))$.



(a) No element remaining.

(b) X_3 and X_5 remain

(c) X_5 remains

Figure 5.1: Orbits in the generalized orbit $\{X_1, X_2, X_3, X_4, X_5\}$ with various choices of representatives

5.5.2 Full sub-orbitopes

Given $X \in \mathscr{X}$ and sets $R \subset \{1,...,m\}$ and $C \subset \{1,...,n\}$, we consider sub-matrix (R,C) of X, denoted by $X(R,C)$, obtained by considering columns C of X on rows R only. A symmetry group is the *sub-symmetric group* with respect to (R,C) if it is the set of all permutations of the columns of $X(R,C)$. If $\mathscr{G}_Q$ is the sub-symmetric group with respect to (R,C) then subset Q is said to be *sub-symmetric* with respect to (R,C).

In this section, we generalize the notion of full orbitope in order to account for sub-symmetries arising in sub-symmetric solution subsets of ILP (5.3).

Consider a set $\mathbb{S}$ of solution subsets Q_i, $i \in \{1,...,s\}$, such that for each $i \in \{1,...,s\}$, Q_i is sub-symmetric with respect to (R_i, C_i).

For each orbit O_k^i, $k \in \{1,...,o_i\}$, let its representative $X_k^i \in O_k^i$ be such that sub-matrix $X_k^i(R_i,C_i)$ is lexicographically maximal, *i.e.*, its columns are lexicographically non-increasing. Such X_k^i is said to be the *lex-max* of orbit O_k^i with respect to (R_i,C_i).

Lemma 5.3. *The set of representatives* $\{X_k^i,\ k \in \{1,...,o_i\},\ i \in \{1,...,s\}\}$ *is orbit-compatible.*

Proof. In order to prove that this set of representatives is orbit-compatible, we prove that there exists a generalized representative of each generalized orbit.

First consider the following row-wise ordering of matrix entries: $(1,1)$, $(1,2)$, ..., $(1,n)$, $(2,1)$, $(2,2)$, ..., $(2,n)$, ..., (m,n). We define an ordering $\succ_M$ of the matrices such that for two matrices A

and B, $A \succ_M B$ if $A(i,j) > B(i,j)$, with (i,j) the first position, with respect to the given ordering of matrix entries, where A and B differ.

For a given solution matrix $X \in \mathscr{X}$, we propose an algorithm computing a generalized representative of the generalized orbit of X. First set $X^0 = X$. At iteration k of the algorithm, there are two cases. In the first case, there exists $i \in \{1,...,s\}$ such that $X^k \in Q_i$ and sub-matrix $X^k(R_i, C_i)$ is not lexicographically maximal, *i.e.*, there exists a column $j \in C_i$ such that $X^k(R_i, \{j\}) \prec X^k(R_i, \{j+1\})$. In this case, X^{k+1} is set to X^k, except that columns j and $j+1$ of sub-matrix $X^k(R_i, C_i)$ are transposed. Otherwise in the second case, the algorithm stops. The claim is that this algorithm stops at some iteration K, and corresponding matrix X^K is a generalized representative of the generalized orbit of X. Note that at each iteration k, $X^k \succ_M X^{k-1}$. As matrices X^k take values in a finite set, there exists an iteration K at which the algorithm stops. By construction, matrix X^K is in the generalized orbit of X, and for each $i \in \{1,...,s\}$ such that $X \in Q_i$, sub-matrix (R_i, C_i) of X is lexicographically maximal. It is thus a generalized representative of the generalized orbit of X. ■

The *full sub-orbitope* $\mathscr{P}_{sub}(\mathbb{S})$ associated to $\mathbb{S}$ is the convex hull of binary matrices X such that for each $i \in \{1,...,s\}$, if $X \in Q_i$ then the columns of $X(R_i, C_i)$ are lexicographically non-increasing. In particular, $\mathscr{P}_{sub}(\mathbb{S})$ contains the generalized representatives of each generalized orbit $\mathscr{O}$.Note that the full sub-orbitope generalizes the full orbitope, in the sense that for $s = 1$, $\mathbb{S} = \{Q_1\}$, $Q_1 = \mathscr{X}$, $\mathscr{G}_{Q_1} = \mathfrak{S}_n$ and $(R_1, C_1) = (\{1,...,m\}, \{1,...,n\})$, $\mathscr{P}_0(m,n) \cap \mathscr{X} = \mathscr{P}_{sub}(\mathbb{S}) \cap \mathscr{X}$.

5.5.3 Sub-symmetries in the MUCP

In the MUCP, there are also other sources of symmetry, arising from the possibility of permuting some sub-columns of matrices x^h. For example, consider two identical units. Suppose at some time period t, these two units are down and ready to start up. Then their plans after t can be permuted, even if they do not have the same up/down plan before t.

More precisely, a unit $j \in \mathscr{N}$ is *ready to start up* at time $t \in \{1,...,T\}$ if and only if $\forall t' \in \{t - \ell^j, ..., t-1\}$, $x_{t',j} = 0$. Similarly, a unit $j \in \mathscr{N}^k$ is *ready to shut down* at time $t \in \{1,...,T\}$ if and only if $\forall t' \in \{t - L^j, ..., t-1\}$, $x_{t',j} = 1$.

Note that sub-symmetries, defined in Section 5.5, appear in the symmetry groups of the subproblems associated to the B&B nodes. In practice, this is not exploited in [70], where the symmetries considered at each node are all contained in the symmetry group $\mathscr{G}$ of the original problem.

Conclusion

We introduce a framework to deal with symmetries arising from multiple (sub-)symmetry group. We show that all-column-permutation symmetries and sub-symmetries can be handled by restricting the feasible set to the set of lex-max representatives. As a perspective, it would be

interesting to prove that this set of representatives contains a unique generalized representative for each generalized orbit.

In the two following chapters, we devise symmetry-breaking techniques suited to account for sub-symmetries. Chapter 6 introduces an orbitopal fixing algorithm for the full (sub-)orbitope. A general framework to build sub-symmetry-breaking inequalities for sub-symmetric solution subsets is given in Chapter 7.

CHAPTER 6

ORBITOPAL FIXING FOR THE FULL (SUB-)ORBITOPE

Orbitopal fixing introduced in [41] is particularly interesting to break symmetries as it is both flexible and full symmetry-breaking. Moreover, no additional inequalities need to be appended to the formulation, thus it does not increase the size of the LP solved at each node. The authors of [41] have proved that for any face F of C^d, the sets $I_0^\star$ and $I_1^\star$ defining $\mathrm{Fix}_F(P)$ can be characterized when P is a partitioning or a packing orbitope.

There are many problems whose symmetry group $\mathscr{G}$ is the set of all column permutations among given subsets of columns of the x matrix, but whose search space cannot be restricted to a partitioning or a packing orbitope. For example, the UCP with identical units is such that the plans of the units, *i.e.*, the columns, can be permuted in any solution, but there is no general restriction on the number of ones on each row t of matrix X^h, corresponding to the number of type h units up at time t.

As detailed in Sections 5.2.4 and 5.3.2, the authors in [70] propose to break these all-column-permutation symmetries using MOB, *i.e.*, by branching on a disjunction that fixes a larger number of variables than the classical disjunction $x_{i,j} = 0 \vee x_{i,j} = 1$. If the use of MOB removes a large number of symmetries, it is only partial symmetry-breaking. The only way to ensure that MOB will remove all non-representative solutions is to use it alongside with a branching rule that restricts the choice of the variables to be branched on.

We explore a different approach, where, at each node, orbitopal fixing for the full orbitope is used to fix some of the remaining variables left free by branching. At a given node a, once some variables have been fixed by branching, we restrict the solution at node a to be in the full orbitope by setting to 0 (resp. to 1) variables that would yield a non-lexicographically ordered solution if

set to 1 (resp. to 0). This approach preserves flexibility as the choice of the branching disjunctions and variables remains totally free.

In this chapter, we propose an orbitopal fixing algorithm for the full orbitope, by characterizing sets $I_0^\star$ and $I_1^\star$ corresponding to the fixing of the full orbitope at (I_0, I_1). We introduce a dynamic variant of this algorithm where the lexicographical order follows the branching decisions occurring along the B&B search.

For each sub-symmetry group containing a symmetric group acting on the columns of a given submatrix, we show that our orbitopal fixing algorithm can be performed to fix variables in the corresponding full (sub-)orbitope. Experimental results on MUCP instances are presented.

The results proposed in this chapter have been published in [10].

6.1 Intersection with the full orbitope

For convenience, the full orbitope $\mathscr{P}_0(m,n)$ is denoted by P_O in this section. Given a face F of $[0,1]^{(m,n)}$ defined by sets (I_0, I_1), we will characterize the sets $I_0^\star$ and $I_1^\star$ defining the fixing $\text{Fix}_F(P_O)$ of the full orbitope at F. Note that face F can be chosen arbitrarily.

We first define $F(P_O)$-minimality and $F(P_O)$-maximality, which are key properties for matrices. Namely we will see that each column j of an $F(P_O)$-minimal (resp. $F(P_O)$-maximal) matrix is the lexicographically lowest (resp. greatest) possible j^{th} column of any binary matrix $X \in P_O \cap F \cap \{0,1\}^{(m,n)}$.

For any matrix X, the j^{th} column of X is denoted by $X(j)$ and the entry at row i, column j by $X(i,j)$.

Definition 6.1. For a given face F of $[0,1]^{(m,n)}$, a matrix X is said to be $F(P_O)$-minimal (resp. $F(P_O)$-maximal) if $X \in P_O \cap F \cap \{0,1\}^{(m,n)}$ and for any matrix $Y \in P_O \cap F \cap \{0,1\}^{(m,n)}$, $X(j)$ is lexicographically less (resp. greater) than or equal to $Y(j)$, $\forall j \in \{1,...,n\}$, *i.e.*, $X(j) \preceq Y(j)$ (resp. $X(j) \succeq Y(j)$) $\forall j \in \{1,...,n\}$.

The section is organized as follows.

1. Two sequences of matrices $(\underline{M}^j)_{j\in\{1,...,n\}}$ and $(\overline{M}^j)_{j\in\{1,...,n\}}$ are introduced, such that matrices $\underline{M}^1$ and $\overline{M}^n$ will respectively be $F(P_O)$-minimal and $F(P_O)$-maximal.
2. Sets $I_1^\star$ and $I_0^\star$ are determined from $\underline{M}^1$ and $\overline{M}^n$.

We now introduce some definitions. Some matrices considered in this section are partial matrices in the sense that their entries can take values in the set $\{0,1,\times\}$, where $\times$ represents a free variable. A given partial matrix M of size (m,n) is fully given by the pair (S_0,S_1) of index subsets such that the indices corresponding to a 0-entry in M are in subset S_0 and the indices corresponding to a 1-entry in M are in subset S_1. The remaining indices $\{1,...,m\}\times\{1,....,n\}\backslash(S_0 \cup S_1)$ correspond to free variables in M.

For a given column $j \in \{1,...,n-1\}$, the following definitions are useful to compare columns j and $j+1$ of matrix M.

Definition 6.2.

- A row $i \in \{1,...,m\}$ is said to be *j-fixed*, for a given $j < n$, if $M(i,j) \neq \times$ and $M(i,j+1) \neq \times$ and $M(i,j) \neq M(i,j+1)$.

  Let $i_f(M,j)$ be the smallest j-fixed row in $\{1,...,m\}$, if such a row exists, and $m+1$ otherwise.

- A row i is said to be *j-discriminating*, for a given $j < n$, if $M(i,j) \neq 0$ and $M(i,j+1) \neq 1$.

  Let $i_d(M,j)$ be the largest j-discriminating row in $\{1,...,i_f(M,j)\}$ if such a row exists, and 0 otherwise.

Example 6.1. *To illustrate, consider matrix M' defined by pair (S'_0,S'_1), with $S'_0 = \{(4,1),(3,2),(5,2)\}$ and $S'_1 = \{(2,1),(5,1),(4,2),(1,3),(2,3)\}$:*

$$M' = \begin{bmatrix} \times & \times & 1 \\ 1 & \times & 1 \\ \times & 0 & \times \\ 0 & 1 & \times \\ 1 & 0 & \times \end{bmatrix}$$

Only rows 4 and 5 are 1-fixed. Hence $i_f(M',1) = 4$. There is no 2-fixed row, so $i_f(M',2) = 6$. Rows 1, 2, 3 and 5 are 1-discriminating, hence $i_d(M',1) = 3$. Only row 4 is 2-discriminating then $i_d(M',2) = 4$.

6.1.1 Two matrix sequences

We propose an algorithm constructing a sequence of matrices $(\underline{M}^j)_{j\in\{1,...,n\}}$ (resp. $(\overline{M}^j)_{j\in\{1,...,n\}}$) of size (m,n). For each j, matrix $\underline{M}^j$ (resp. $\overline{M}^j$) will be derived from pair $(\underline{S}_0^j,\underline{S}_1^j)$ (resp. $(\overline{S}_0^j,\overline{S}_1^j)$). Matrices $\underline{M}^1$ and $\overline{M}^n$ will respectively be $F(P_O)$-minimal and $F(P_O)$-maximal if $\mathrm{Fix}_F(P_O)$ is non-empty. Otherwise, they will be arbitrarily defined by the sets $S_0^\emptyset = \{(1,1)\}$, $S_1^\emptyset = \{1,....,m\} \times \{1,...,n\} \backslash S_0^\emptyset$.

The key idea for the construction of matrix sequence $(\underline{M}^j)_{j\in\{1,...,n\}}$ is the following. For $j = n$, matrix $\underline{M}^n$ is defined by pair (I_0,I_1), except that each free variable in column n is set to 0. For each $j < n$, matrix $\underline{M}^j$ is defined to be equal to matrix $\underline{M}^{j+1}$, except that free variables in column $\underline{M}^{j+1}(j)$ are set to 0 or 1 in matrix $\underline{M}^j$. This is done by propagating values from column $j+1$, so that column j is minimum among all columns greater than or equal to column $j+1$. Note that in matrix $\underline{M}^j$, there are no remaining free variables in columns $\{j,...,n\}$.

The construction of sequence $(\underline{M}^j)_{j\in\{1,...,n\}}$ is given in Algorithm 3. For $j = n$, matrix $\underline{M}^n$ is defined by pair $(I_0 \cup \{(i,n) \notin I_1\}, I_1)$. For $j < n$, if $i_f(\underline{M}^{j+1},j) = m+1$ then each free variable in

Algorithm 3 Construction of sequence $(\underline{M}^j)_{j\in\{1,\dots,n\}}$ defined by pair $(\underline{S}_0^j, \underline{S}_1^j)_{j\in\{1,\dots,n\}}$

$j \leftarrow n$.
$\underline{S}_1^n \leftarrow I_1$
$\underline{S}_0^n \leftarrow \{(i,n) \notin I_1\} \cup I_0$
for $j = n-1$ to 1 **do**
  $i_f \leftarrow i_f(\underline{M}^{j+1}, j)$
  if $i_f = m+1$ **then**
    $\underline{S}_1^j \leftarrow \underline{S}_1^{j+1} \cup \left\{(i,j) \notin \underline{S}_0^{j+1} \mid (i,j+1) \in \underline{S}_1^{j+1}\right\}$
    $\underline{S}_0^j \leftarrow \underline{S}_0^{j+1} \cup \left\{(i,j) \notin \underline{S}_1^{j+1} \mid (i,j+1) \in \underline{S}_0^{j+1}\right\}$
  else if there is no j-discriminating row $i \in \{1,\dots,i_f\}$ in matrix $\underline{M}^{j+1}$ **then**
    $(\underline{S}_0^{j'}, \underline{S}_1^{j'}) \leftarrow (S_0^{\varnothing}, S_1^{\varnothing})$, $\forall j' \le j$
  else
    $i_d \leftarrow i_d(\underline{M}^{j+1}, j)$
    $\underline{S}_1^j \leftarrow \underline{S}_1^{j+1} \cup \{(i_{ld}, j)\} \cup \left\{(i,j) \notin \underline{S}_0^{j+1} \mid (i,j+1) \in \underline{S}_1^{j+1} \text{ and } i < i_{ld}\right\}$
    $\underline{S}_0^j \leftarrow \underline{S}_0^{j+1} \cup \left\{(i,j) \notin \underline{S}_1^j\right\}$.
  end if
end for

column $\underline{M}^j(j)$ is set such that columns j and $j+1$ are equal. Otherwise, there are two cases. In the first case, $i_f(\underline{M}^{j+1}, j) \le m$ and there is no j-discriminating row $i \in \{1,\dots,i_f\}$ in matrix $\underline{M}^{j+1}$. Then for all $j' \le j$, $(\underline{S}_0^{j'}, \underline{S}_1^{j'})$ is set to $(S_0^{\varnothing}, S_1^{\varnothing})$. In the second case, $i_f(\underline{M}^{j+1}, j) \le m$ and there exists a j-discriminating row $i \in \{1,\dots,i_f\}$ in matrix $\underline{M}^{j+1}$. Let row $i_d = i_d(\underline{M}^{j+1}, j)$. Free variables in column $\underline{M}^j(j)$ are set such that columns j and $j+1$ are equal from row 1 to row $i_d - 1$, and such that row i_d has the form [1, 0] on columns j and $j+1$. Every other free variable in column j is set to 0.

As the definition of sequence $(\overline{M}^j)_{j\in\{1,\dots,n\}}$ is very similar, the corresponding algorithm is omitted. For $j = 1$, matrix $\overline{M}^1$ is defined by pair $(I_0, I_1 \cup \{(i,1) \notin I_0\})$. For $j > 1$, free variables in column $\overline{M}^{j-1}(j)$ are set to 0 or 1 in matrix $\overline{M}^j$ by propagating values from column $j-1$, so that column j is maximum among all columns less than or equal to column $j-1$.

Referring to (S'_0, S'_1) defined in Example 6.1 alongside with matrix M', corresponding matrix sequence $(\underline{M}^k)_{k\in\{1,2,3\}}$ is as follows.

$$\underline{M}^3 = \begin{bmatrix} \times & \times & 1 \\ 1 & \times & 1 \\ \times & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad \underline{M}^2 = \begin{bmatrix} \times & 1 & 1 \\ 1 & 1 & 1 \\ \times & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad \underline{M}^1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

The first 2-fixed row in matrix $\underline{M}^3$ is row 4. Row 4 is also the last 2-discriminating row in matrix $\underline{M}^3$ before row 4. Thus $i_f(M', 2) = 4$, $i_d(M', 2) = 4$ and variables (1,2) and (2,2) in matrix

$\underline{M}^2$ are set to be equal the corresponding values in column $\underline{M}^3(2)$. The first 1-fixed row in matrix $\underline{M}^2$ is row 4. The last 1-discriminating row before row 4 in matrix $\underline{M}^2$ is row $i_d(\underline{M}^2,1)=3$. Since $i_d(\underline{M}^2,1)=3$, entries (1,1) and (3,1) are set to 1 in matrix $\underline{M}^1$. Matrix sequence $(\overline{M}^k)_{k\in\{1,2,3\}}$ is obtained similarly. Finally, for any matrix X in the face defined by (S'_0,S'_1), Theorem 6.1 shows that the following inequalities hold column-wise:

$$\underline{M}^1=\begin{bmatrix}1&1&1\\1&1&1\\1&0&0\\0&1&0\\1&0&0\end{bmatrix}\quad\preceq\quad X\quad\preceq\quad\overline{M}^3=\begin{bmatrix}1&1&1\\1&1&1\\1&0&0\\0&1&1\\1&0&0\end{bmatrix}$$

Theorem 6.1. *If $(\underline{S}^1_0,\underline{S}^1_1)=(S^\varnothing_0,S^\varnothing_1)$ or $(\overline{S}^n_0,\overline{S}^n_1)=(S^\varnothing_0,S^\varnothing_1)$ then $Fix_F(P_O)=\varnothing$. Otherwise matrix $\underline{M}^1$ is $F(P_O)$-minimal and matrix $\overline{M}^n$ is $F(P_O)$-maximal.*

Proof. We will prove that if $(\underline{S}^j_0,\underline{S}^j_1)\neq(S^\varnothing_0,S^\varnothing_1)$, then, $\forall X\in \mathrm{Fix}_F(P_O)$, $\forall j\in\{1,\dots,n\}$, $\underline{M}^j(j)\preceq X(j)$, and otherwise $\mathrm{Fix}_F(P_O)=\varnothing$. A similar proof can be done to obtain the corresponding result for $(\overline{S}^n_0,\overline{S}^n_1)$ and $\overline{M}^j$. The property is proved by induction on decreasing index value $j\in\{1,\dots,n\}$.

For $j=n$, by construction $(\underline{S}^n_0,\underline{S}^n_1)\neq(S^\varnothing_0,S^\varnothing_1)$. Since all $(i,n)\notin I_1$ are set to 0 in matrix $\underline{M}^n$, necessarily $\forall X\in P_O\cap F\cap\{0,1\}^{(m,n)}$, $\underline{M}^n(n)\preceq X(n)$.

Suppose the induction hypothesis holds for $j+1$, with $j<n$. There are two cases: either $(\underline{S}^j_0,\underline{S}^j_1)\neq(S^\varnothing_0,S^\varnothing_1)$ or not.

On the one hand, suppose $(\underline{S}^j_0,\underline{S}^j_1)\neq(S^\varnothing_0,S^\varnothing_1)$. Suppose also there exists $X\in P_O\cap F\cap\{0,1\}^{(m,n)}$ such that $\underline{M}^j(j)\succ X(j)$. Consider the first row i such that columns $X(j)$ and $\underline{M}^j(j)$ are different. As $\underline{M}^j(j)\succ X(j)$, we have $X(i,j)=0$ and $\underline{M}^j(i,j)=1$. By construction, since $(i,j)\notin I_0\cup I_1$ and $\underline{M}^j(i,j)=1$, for all $i'<i$, $\underline{M}^j(i',j)=\underline{M}^j(i',j+1)$. If $\underline{M}^j(i,j+1)=1$, then since $\underline{M}^j(j+1)=\underline{M}^{j+1}(j+1),\underline{M}^{j+1}(j+1)\succ X(j)$. By the induction hypothesis, $X(j+1)\succeq\underline{M}^{j+1}(j+1)$ thus $X(j+1)\succ X(j)$, which contradicts $X\in P_O$. Let now suppose $\underline{M}^j(i,j+1)=0$, then, from the construction of $\underline{M}^j$, row $i_f=i_f(\underline{M}^{j+1},j)$ in matrix $\underline{M}^{j+1}$ has the form [0, 1] on columns j and $j+1$ (otherwise $\underline{M}^j(i,j)$ would have been set to 0). In this case, row i corresponds to the last j-discriminating row of matrix $\underline{M}^{j+1}$ before row i_f. Thus, for each $i'\in\{i+1,i_f-1\}$ such that $(i',j)\notin I_0\cup I_1$, we have $\underline{M}^j(i',j+1)=1$. If for such an i', $X(i',j)=0$ then since $\underline{M}^j(j+1)=\underline{M}^{j+1}(j+1)$, $\underline{M}^{j+1}(j+1)\succ X(j)$. Otherwise, as row i_f in matrix $\underline{M}^{j+1}$ has the form [0, 1] on columns j and $j+1$, it follows $(i_f,j)\in I_0$, thus $X(i_f,j)=0$. Consequently $\underline{M}^{j+1}(j+1)\succ X(j)$ holds too. By the induction hypothesis, $X(j+1)\succeq\underline{M}^{j+1}(j+1)$ thus we reach the same contradiction.

On the other hand, suppose $(\underline{S}^j_0,\underline{S}^j_1)=(S^\varnothing_0,S^\varnothing_1)$, consider the following two cases: If $(\underline{S}^{j+1}_0,\underline{S}^{j+1}_1)=(S^\varnothing_0,S^\varnothing_1)$ then by the induction hypothesis, $\mathrm{Fix}_F(P_O)=\varnothing$. Otherwise, $(\underline{S}^{j+1}_0,\underline{S}^{j+1}_1)\neq(S^\varnothing_0,S^\varnothing_1)$. Recall $i_f=i_f(\underline{M}^{j+1},j)$. Then, by construction of matrix $\underline{M}^j$, row i_f of matrix $\underline{M}^{j+1}$ has the form [0, 1] on columns j and $j+1$ and there is no row $i\in\{1,\dots,i_f-1\}$ in matrix $\underline{M}^{j+1}$ which is j-discriminating. As column $j+1$ is completely fixed in matrix $\underline{M}^{j+1}$, each row $i\in\{1,\dots,i_f-1\}$ of

matrix $\underline{M}^{j+1}$ has one the following forms on columns j and $j+1$: [1, 1] or [0, 0] or [×, 1]. Therefore, if $\text{Fix}_F(P_O)$ were not empty, then $P_O \cap F \cap \{0,1\}^{(m,n)} \neq \varnothing$ and for any $X \in P_O \cap F \cap \{0,1\}^{(m,n)}$, even if $X(i,j) = 1$ for each $(i,j) \notin I_0 \cup I_1$, $\underline{M}^{j+1}(j+1) \succ X(j)$ would hold. By the induction hypothesis, $X(j+1) \succeq \underline{M}^{j+1}(j+1)$ thus $X(j+1) \succ X(j)$, which contradicts $X \in P_O$. ■

6.1.2 Determining $I_0^\star$ and $I_1^\star$

In case $\text{Fix}_F(P_O) \neq \varnothing$, sets $I_0^\star$ and $I_1^\star$ can be characterized using $F(P_O)$-minimal and $F(P_O)$-maximal matrices $\underline{M}^1$ and $\overline{M}^n$ as follows. For each $j \in \{1,...,m\}$, consider row i_j, the first row at which columns $\underline{M}^1(j)$ and $\overline{M}^n(j)$ differs, defined as:

$$i_j = \min\left\{i \in \{1,...,m\} \mid \underline{M}^1(i,j) \neq \overline{M}^n(i,j)\right\}$$

If columns $\underline{M}^1(j)$ and $\overline{M}^n(j)$ are equal, then i_j is arbitrarily set to $m+1$. By definition of $F(P_O)$-minimal and $F(P_O)$-maximal matrices, $\underline{M}^1(i_j,j) < \overline{M}^n(i_j,j)$. Note that since for all $(i,j) \in I_0$ (resp. I_1), $\underline{M}^1(i,j) = \overline{M}^n(i,j) = 0$ (resp. 1), it follows that (i_j,j) is a free variable *i.e.*, $(i_j,j) \notin I_0 \cup I_1$.

Theorem 6.2. *$Fix_F(P_O)$, if non-empty, is given by sets $I_0^\star = I_0 \cup I_0^+$ and $I_1^\star = I_1 \cup I_1^+$, where*

$$I_0^+ = \left\{(i,j) \notin I_0 \cup I_1 \mid i < i_j \text{ and } \overline{M}^n(i,j) = 0\right\}, \qquad I_1^+ = \left\{(i,j) \notin I_0 \cup I_1 \mid i < i_j \text{ and } \underline{M}^1(i,j) = 1\right\}.$$

Proof. ($\Longrightarrow$) We prove that $I_0^+ \subset I_0^\star$ and $I_1^+ \subset I_1^\star$. Let us suppose the opposite: $I_0^+ \not\subset I_0^\star$ or $I_1^+ \not\subset I_1^\star$. Let $(i,j) \in (I_0^+ \backslash I_0^\star) \cup (I_1^+ \backslash I_1^\star)$. Consider $i_0 = \min\{i' \mid (i',j) \in (I_0^+ \backslash I_0^\star) \cup (I_1^+ \backslash I_1^\star)\}$. Suppose $(i_0,j) \in I_0^+ \backslash I_0^\star$. The proof is similar if we suppose $(i_0,j) \in I_1^+ \backslash I_1^\star$. As $(i_0,j) \notin I_0^\star$, there exists $X \in P_O \cap F \cap \{0,1\}^{(m,n)}$ such that $X(i_0,j) = 1$. As $(i_0,j) \in I_0^+$, $\overline{M}^n(i_0,j) = 0$. If for all $i' < i_0$, $X(i',j) \geq \overline{M}^n(i',j)$ then the following would hold: $X(j) \succ \overline{M}^n(j)$, contradicting the fact that $\overline{M}^n$ is $F(P_O)$-maximal. Thus, there exists a row $i_1 < i_0$ such that $\overline{M}^n(i_1,j) = 1$ and $X(i_1,j) = 0$. Note that as $(i_0,j) \in I_0^+$, $i_0 < i_j$, and thus $i_1 < i_j$, which implies $\underline{M}^1(i_1,j) = 1$. Thus $(i_1,j) \in I_1^+$. However, $(i_1,j) \notin I_1^\star$ because $X \in P_O \cap F \cap \{0,1\}^{(m,n)}$ and $X(i_1,j) = 0$. The contradiction comes from the fact that $i_1 < i_0$ and $i_1 \in \{i' \mid (i',j) \in (I_0^+ \backslash I_0^\star) \cup (I_1^+ \backslash I_1^\star)\}$. This proves $I_0^+ \subset I_0^\star$ and $I_1^+ \subset I_1^\star$, thus $I_0 \cup I_0^+ \subset I_0^\star$ and $I_1 \cup I_1^+ \subset I_1^\star$.

($\Longleftarrow$) We prove $I_0^\star \subset I_0 \cup I_0^+$ and $I_1^\star \subset I_1 \cup I_1^+$. It suffices to show that given $(i,j) \notin I_0^\star \cup I_1^\star$, there exists a solution $X_0 \in P_O \cap F \cap \{0,1\}^{(m,n)}$ such that $X_0(i,j) = 0$ and a solution $X_1 \in P_O \cap F \cap \{0,1\}^{(m,n)}$ such that $X_1(i,j) = 1$. Consider index $(i_j,j) \notin I_0 \cup I_1$. Solution $\underline{M}^1$ is such that $\underline{M}^1(i_j,j) = 0$ and solution $\overline{M}^n$ is such that $\overline{M}^n(i_j,j) = 1$. So if $i = i_j$, the result is proved. Now suppose $i \neq i_j$. Note that for all $i' < i_j$, $\underline{M}^1(i',j) = \overline{M}^n(i',j)$, therefore $(i',j) \in I_0^\star \cup I_1^\star$. Thus $i > i_j$. Consider solutions X_0 and X_1 defined as follows. For each $i' \in \{1,...,m\}$ and $j' \in \{1,...,n\}$,

$$X_0(i',j') = \begin{cases} \overline{M}^n(i',j') & \text{if } j' < j \\ \underline{M}^1(i',j') & \text{if } j' > j \\ \underline{M}^1(i',j') & \text{if } j' = j \text{ and } i' < i_j \\ 1 & \text{if } j' = j \text{ and } i' = i_j \\ 0 & \text{otherwise.} \end{cases} \qquad X_1(i',j') = \begin{cases} \overline{M}^n(i',j') & \text{if } j' < j \\ \underline{M}^1(i',j') & \text{if } j' > j \\ \underline{M}^1(i',j') & \text{if } j' = j \text{ and } i' < i_j \\ 0 & \text{if } j' = j \text{ and } i' = i_j \\ 1 & \text{otherwise.} \end{cases}$$

Recall that $\overline{M}^n(i_j,j)=1$ and $\underline{M}^1(i_j,j)=0$, therefore $\overline{M}^n(j)\succeq X_0(j)\succ X_1(j)\succ \underline{M}^1(j)$. As $\overline{M}^n$ and $\underline{M}^1\in P_O$, $\overline{M}^n(j-1)\succeq\overline{M}^n(j)$ and $\underline{M}^1(j)\succeq\underline{M}^1(j+1)$. Thus X_1 and X_0 are also in $P_O\cap F\cap\{0,1\}^{(m,n)}$ and are such that $X_1(i,j)=1$ and $X_0(i,j)=0$. This concludes the proof. ■

To illustrate, consider matrices $\underline{M}^1$ and $\overline{M}^3$ from Example 6.1. Here the rows i_j are respectively $i_1=6$, since $\underline{M}^1(1)=\overline{M}^3(1)$, $i_2=6$ and $i_3=4$. The corresponding sets I_0^+ and I_1^+ are $I_0^+=\{(3,3)\}$ and $I_1^+=\{(1,1),(3,1),(1,2),(2,2)\}$. Note that indices $(4,3)$ and $(5,3)$ are neither in I_0^+ nor in I_1^+ because they belong to rows greater than or equal $i_3=4$. The associated variables cannot be fixed, even though variable $x(5,3)$ is set to 0 in $\underline{M}^1$ and $\overline{M}^3$.

Matrices $\underline{M}^1$ and $\overline{M}^n$ can be computed in $O(mn)$ time, since at each iteration $j\in\{1,...,n\}$ of Algorithm 3, i_f and i_{ld} can be computed in $O(m)$ time. Once matrices $\underline{M}^1$ and $\overline{M}^n$ are known, sets $I_0^\star$ and $I_1^\star$ can be computed in $O(mn)$ time as well. It follows:

Theorem 6.3. *$Fix_F(P_O)$ can be computed in linear time $O(mn)$.*

6.2 Static and dynamic orbitopal fixing

So far, the considered lexicographical order on the columns was defined with respect to order $1,....,m$ on the rows. In this section, we define a *static orbitopal fixing* algorithm for the full orbitope, which relies on this lexicographical order. We also define a *dynamic orbitopal fixing* algorithm for the full orbitope, where the lexicographical order is defined with respect to an order on the rows determined by the branching decisions in the B&B tree. Interestingly these static and dynamic variants of the orbitopal fixing algorithm can be used also for the full sub-orbitope case. It is worth noting that this orbitopal fixing algorithm based on our intersection result from Section 6.1 performs all possible variable fixings (with respect to the full (sub-)orbitope) as early as possible in the B&B tree.

6.2.1 Static orbitopal fixing

When solving ILP (5.3) with B&B, static orbitopal fixing can be performed at the beginning of each node processing in the B&B tree, in order to ensure that any enumerated solution x in the B&B tree is such that $x\in\mathscr{P}_0(m,n)$, assuming the lexicographical order is a priori given.

The *static orbitopal fixing* algorithm at node a is the following:

- Set $I_0=B_0^a$, $I_1=B_1^a$, where B_0^a (resp. B_1^a) is the index set of variables previously fixed to 0 (resp. 1).
- Compute matrices $\underline{M}^1$ and $\overline{M}^n$ using Algorithm 1.
- If $\underline{M}^1$ or $\overline{M}^n$ is defined by pair $(S_0^\varnothing,S_1^\varnothing)$, prune node a. Otherwise determine I_0^+ and I_1^+ using Th. 6.2.

- Fix variable $x_{i,j}$ to 0, for each $(i,j) \in I_0^+$.

- Fix variable $x_{i,j}$ to 1, for each $(i,j) \in I_1^+$.

From Theorem 6.2, the pair $(I_0^\star, I_1^\star) = (I_0 \cup I_0^+, I_1 \cup I_1^+)$ defines $\mathrm{Fix}_{F(a)}(\mathscr{P}_0(m,n))$, where $F(a)$ is the hypercube face given by (B_0^a, B_1^a) at each node a of a B&B tree. Thus the following result can be directly derived.

Theorem 6.4. *Let τ be a B&B tree of ILP (5.3), in which static orbitopal fixing is performed, and the branching rule is arbitrary. For each solution orbit $\mathscr{O}$ of ILP (5.3), there is exactly one solution of $\mathscr{O}$ enumerated in B&B tree τ.*

From Theorem 6.3, the static orbitopal fixing algorithm is in $O(mn)$ time at each node of the B&B tree.

6.2.2 Dynamic orbitopal fixing

In the previous sections, the lexicographical order on the columns of an $m \times n$ binary matrix was defined with respect to the order $1,...,m$ on the rows. Note that this order is arbitrary, and thus the definition of the lexicographical order can be extended for any ordering of the m rows. Namely, considering a bijection $\phi : \{1,...,m\} \to \{1,...,m\}$, column c is lexicographically greater than or equal to a column c', with respect to ordering ϕ, if there exists $i \in \{1,....,m-1\}$ such that $\forall i' \leq i$, $y_{\phi(i')} = z_{\phi(i')}$ and $y_{\phi(i)+1} > z_{\phi(i)+1}$.

Dynamic fixing is to perform, at any node a of the B&B tree, orbitopal fixing with respect to reorderings ϕ_a of the row indices, defined by the branching decisions leading to node a. The idea of pruning the B&B tree with respect to an order defined by the branching process has been introduced by Margot [62].

As a first step, suppose at each node a of the B&B tree, the branching disjunction has the form

$$x_{i_a,j_a} = 0 \quad \vee \quad x_{i_a,j_a} = 1. \tag{6.1}$$

Dynamic orbitopal fixing is to perform orbitopal fixing on row set $I_a = \{\, \phi_a(1), \phi_a(2), ..., \phi_a(|I_a|) \,\}$ at each node a, where lexicographical ordering ϕ_a and I_a are defined recursively as follows.

$$\text{If } a \text{ is the root, then } \begin{cases} I_a = \{i_a\} \\ \phi_a(1) = i_a \end{cases}, \text{ otherwise } \begin{cases} I_a = I_b \cup \{i_a\} & \\ \phi_a(i) = \phi_b(i) & \forall i \in \{1,...,|I_b|\}. \\ \phi_a(|I_b|+1) = i_a & \text{if } i_a \notin I_b, \\ \text{where } b \text{ is the father of } a. & \end{cases}$$

Note that an arbitrary branching rule used alongside with an arbitrary ordering may lead to the removal of every optimal solution from the B&B tree. The following theorem shows that the use of branching rule (6.1) and ordering ϕ_a preserves an optimal solution in the B&B tree.

Theorem 6.5. *Let τ be a B&B tree of ILP (5.3), in which dynamic orbitopal fixing is performed and branching disjunctions have the form (6.1). For each solution orbit $\mathcal{O}$ of ILP (5.3), there is exactly one solution of $\mathcal{O}$ enumerated in B&B tree τ.*

Proof. The sketch of the proof is to produce a solution $X \in \mathcal{O}$ and prove that X is the only solution of $\mathcal{O}$ which is enumerated in τ.

First consider the branching disjunction at the root node a_r: $(x_{i_0,j_0} = 0 \quad \vee \quad x_{i_0,j_0} = 1)$. Then $\phi_{a_r}(1) = i_0$. Let n_{i_0} be the number of 1-entries on row i_0 of any matrix $X \in \mathcal{O}$. Since dynamic orbitopal fixing is enforced in τ, any solution enumerated by τ must be lexicographically non-increasing with respect to ϕ_{a_r}. Then, as row i_0 is the first row with respect to the lexicographical order ϕ_{a_r}, any $X \in \mathcal{O}$ enumerated by the B&B tree will be such that:

$$X(i_0,j) = 1,\ \forall j \in \{1,\dots,n_{i_0}\} \qquad \text{and} \qquad X(i_0,j) = 0,\ \forall j \in \{n_{i_0}+1,\dots,n\}$$

Note that if $j_0 \le n_{i_0}$ (resp. $j_0 > n_{i_0}$) then any $X \in \mathcal{O}$ enumerated by τ is such that $X(i_0,j_0) = 1$ (resp. $X(i_0,j_0) = 0$). Thus there is no solution of $\mathcal{O}$ in the branch $x_{i_0,j_0} = 0$ (resp. $x_{i_0,j_0} = 1$).

Suppose w.l.o.g. that $j_0 \le n_{i_0}$, so the node considered is b_1, the son of a_r such that $x_{i_0,j_0} = 1$. Consider the branching disjunction at node b_1: $(x_{i_1,j_1} = 0 \quad \vee \quad x_{i_1,j_1} = 1)$. If $i_1 = i_0$ then, by the same arguments as at the root node, there is exactly one branch in which all elements of $\mathcal{O}$ are enumerated, and this branch can be easily determined. Otherwise, since $i_1 \neq i_0$, then by construction, $\phi_{b_1}(1) = i_0$ and $\phi_{b_1}(2) = i_1$. Let $n^1_{i_1}$ (resp. $n^0_{i_1}$) be the number of columns j such that $X(i_0,j) = 1$ (resp. $X(i_0,j) = 0$) and $X(i_1,j) = 1$. Since row i_1 is second with respect to lexicographical order ϕ_{b_1}, any $X \in \mathcal{O}$ enumerated by the B&B tree will be such that:

$$\begin{cases} X(i_1,j) = 1 & \forall j \in \{1,\dots,n^1_{i_1}\} \cup \{n_{i_0}+1,\dots,n_{i_0}+n^0_{i_1}\} \\ X(i_1,j) = 0 & \forall j \in \{n^1_{i_1}+1,\dots,n_{i_0}\} \cup \{n_{i_0}+n^0_{i_1}+1,\dots,n\} \end{cases}$$

Thus, all $X \in \mathcal{O}$ enumerated by τ have the same value v in entry (i_1,j_1), and this value can be determined, as previously, by finding to which of the sets $\{1,\dots,n^1_{i_1}\}$, $\{n^1_{i_1}+1,\dots,n_{i_0}\}$, $\{n_{i_0}+1,\dots,n_{i_0}+n^0_{i_1}\}$, $\{n_{i_0}+n^0_{i_1}+1,\dots,n\}$ does index j_1 belong. Therefore, since for all $X \in \mathcal{O}$ enumerated by τ, $X(i_1,j_1) = v$, there is exactly one branch $(x_{i_1,j_1} = v)$ in which any $X \in \mathcal{O}$ is enumerated. This process can be repeated until a leaf node a_l is reached. At that point, all entries of X are determined. By construction, X is the only element of $\mathcal{O}$ enumerated by τ, since at each node we considered, there was always a unique branch leading to all elements of $\mathcal{O}$. ■

Now suppose the branching disjunction at each node a is arbritrary. The latter result can be extended to show that dynamic orbitopal fixing can also be used in this case. For each node a, consider a branching disjunction of the form:

$$\sum_{i \in \mathcal{R}_a} \sum_{j=1}^{p} \lambda^i_a x_{i,j} \le k \quad \vee \quad \sum_{i \in \mathcal{R}_a} \sum_{j=1}^{p} \lambda^i_a x_{i,j} > k. \tag{6.2}$$

where $\mathcal{R}_a = \{r_{a,1},\dots,r_{a,p}\} \subset \{1,\dots,m\}$.

A new lexicographical ordering $\widetilde{\phi}_a$ taking into account every row involved in disjunction (6.2) must be defined at each node a. Namely, row subset $\widetilde{I}_a \subset \{1,...,m\}$ and bijection $\widetilde{\phi}_a : \{1,...,|\widetilde{I}_a|\} \to \widetilde{I}_a$ are as follows.

$$\text{If } a \text{ is the root, then } \begin{cases} \widetilde{I}_a = \mathscr{R}_a \\ \widetilde{\phi}_a(k) = r_{a,k}, \quad k \in \{1,...,p\} \end{cases}, \text{ otherwise } \begin{cases} \widetilde{I}_a = \widetilde{I}_b \cup \mathscr{R}_a \\ \widetilde{\phi}_a(i) = \widetilde{\phi}_b(i), \forall i \in \{1,...,|\widetilde{I}_b|\} \\ \widetilde{\phi}_a(|\widetilde{I}_b|+k) = r'_{a,k}, \forall k \in \{1,...,p'\} \\ \text{where } b \text{ is the father of } a \\ \text{and } \{r'_{a,1},...,r'_{a,p'}\} = \mathscr{R}_a \setminus \widetilde{I}_b. \end{cases}$$

6.2.3 Orbitopal fixing in the full sub-orbitope

Consider a collection $\mathbb{S}$ of sub-symmetric solution subsets $Q_i \subset \mathscr{X}$, $i \in \{1,...,s\}$ with respect to (R_i, C_i). Static (resp. dynamic) orbitopal fixing can be performed for $\mathscr{P}_{sub}(\mathbb{S})$ at each node a of the B&B tree as follows. Consider $\mathfrak{I}_a \subset \{1,...,s\}$ such that for each $i \in \mathfrak{I}_a$, each solution x to the sub-problem at node a is in Q_i. The idea is to apply static (resp. dynamic) orbitopal fixing to the submatrix $X(R_i, C_i)$, for each $i \in \mathfrak{I}_a$.

By Lemma 5.3 the representatives associated with the natural lexicographical order are orbit-compatible. Consequently, static orbitopal fixing for $\mathscr{P}_{sub}(\mathbb{S})$ does not change the optimal value returned by the B&B process. Lemma 5.3 can directly be adapted to the case when the representatives are associated to a lexicographical order defined by arbitrary row-order ϕ, and the proof of Theorem 6.5 can be slightly modified to show that dynamic orbitopal fixing for $\mathscr{P}_{sub}(\mathbb{S})$ is also valid.

Referring to the vocabulary of Section 5.2.5, the orbitopal fixing algorithm we devised for the full orbitope is simultaneous. In the full sub-orbitope case, the fixing process is iterated over all the submatrices associated to the sub-symmetric groups, and therefore it corresponds to sequential fixing.

6.3 Orbitopal fixing for the MUCP

For each time period $t \in \{1,...,\mathscr{T}\}$ and subset $\mathfrak{N} \subset \mathscr{N}_h$, $h \in \{1,...,H\}$ of identical units, consider set $\mathbb{S}_{MUCP}$ containing the following subsets of $\mathscr{X}_{MUCP}$:

$$\overline{Q}^t_{\mathfrak{N}} = \{X \in \mathscr{X}_{MUCP} \mid X(t',j) = 0,\ \forall t' \in \{t-\ell^j,...,t-1\},\ \forall j \in \mathfrak{N}\}$$

$$\underline{Q}^t_{\mathfrak{N}} = \{X \in \mathscr{X}_{MUCP} \mid X(t',j) = 1,\ \forall t' \in \{t-L^j,...,t-1\},\ \forall j \in \mathfrak{N}\}$$

Note that at each node a of the tree, it is easy to find the sets $\overline{Q}^t_{\mathfrak{N}}$ and $\underline{Q}^t_{\mathfrak{N}}$, $t \in \{1,...,\mathscr{T}\}$, $\mathfrak{N} \subset \mathscr{N}_h$, $h \in \{1,...,H\}$, to which any solution of the subproblem associated to node a belongs.

Indeed, for each time period t and for each unit i down (resp. up) at time t, it is possible to know how long unit i has been down (resp. up), and thus whether unit i is ready to start up (resp. shut down) or not. If we denote by $\mathfrak{N}^u_{t,h}$ (resp. $\mathfrak{N}^d_{t,h}$) the set of type h units which are ready to start up (resp. shut down) at time t, then all solutions at node a are in sets $\overline{Q}^t_{\mathfrak{N}^u_{t,h}}$ and $\underline{Q}^t_{\mathfrak{N}^d_{t,h}}$.

Let $\mathscr{G}_{\overline{Q}^t_{\mathfrak{N}}}$ and $\mathscr{G}_{\underline{Q}^t_{\mathfrak{N}}}$ be the sub-symmetry groups associated to $\overline{Q}^t_{\mathfrak{N}}$ and $\underline{Q}^t_{\mathfrak{N}}$, $t \in \{1,...,\mathscr{T}\}$, $\mathfrak{N} \subset \mathscr{N}_h$, $h \in \{1,...,H\}$. Note that groups $\mathscr{G}_{\overline{Q}^t_{\mathfrak{N}}}$ and $\mathscr{G}_{\underline{Q}^t_{\mathfrak{N}}}$ contain the sub-symmetric group associated to the submatrix defined by rows and columns $(\{t,...,T\},\mathfrak{N})$. The corresponding full sub-orbitope is denoted by $\mathscr{P}_{sub}(\mathbb{S}_{MUCP})$.

As the production plans of identical units can be permuted, each variable matrix X^h can be restricted to be in the full orbitope $\mathscr{P}_0(T,n_h)$. More generally we have seen in Section 5.5.3 that variable matrix X can be restricted to be in the full sub-orbitope $\mathscr{P}_{sub}(MUCP)$

The fixing strategies developed in Sections 6.2.1 and 6.2.2 can thus be applied to fix variables in each matrix X^h, in order to enumerate only solutions with lexicographically maximal X^h. These strategies can also be applied to restrict the feasible set to the full sub-orbitope $\mathscr{P}_{sub}(MUCP)$.

The possible approaches are the following:

- Static orbitopal fixing (SOF) for the full orbitopes $\mathscr{P}_0(T,n_h)$, $h \in \{1,...,H\}$, where the order on the rows is decided before the branching process.
- Dynamic orbitopal fixing (DOF) for the full orbitopes $\mathscr{P}_0(T,n_h)$, $h \in \{1,...,H\}$, where the order on the rows $\tilde{\phi}$ is decided during the branching process, as described in Section 6.2.2.
- Static orbitopal fixing for the full orbitopes $\mathscr{P}_0(T,n_h)$, $h \in \{1,...,H\}$ and for the full sub-orbitope $\mathscr{P}_{sub}(MUCP)$.
- Dynamic orbitopal fixing for the full orbitopes $\mathscr{P}_0(T,n_h)$, $h \in \{1,...,H\}$ and for the full sub-orbitope $\mathscr{P}_{sub}(MUCP)$.

In the static case, the branching decisions are completely free. As stated in Section 6.2.2, the branching decisions remain free in the dynamic case, provided that the corresponding rows are ordered accordingly. In our experiments, we only consider the branching disjunctions of the form $(x_{t,j} = 0 \vee x_{t,j} = 1)$, or $(x_{t,j} - x_{t-1,j} \leq 0 \quad \vee \quad x_{t,j} - x_{t-1,j} = 1)$, *i.e.*, $(u_{t,j} = 0 \vee u_{t,j} = 1)$.

6.4 Experimental results for the MUCP

All experiments were performed using one thread of a PC with a 64 bit Intel Core i7-6700 processor running at 3.4GHz, and 32 GB of RAM memory. The MUCP instances are solved until optimality (defined within 10^{-7} of relative optimality tolerance) or until the time limit of 3600 seconds is reached.

In the following experiments, we compare MUCP resolution methods pairwise using a speed-up indicator. For given approaches m_1 and m_2, the *speed-up* achieved by m_1 with respect to m_2

on a given instance is the ratio $\frac{CPU(m_2)}{CPU(m_1)}$. The *average speed-up*, computed on a set I of p instances, is the geometric mean $(\Pi_{i=1}^{p} s_i)^{\frac{1}{p}}$ of the speed-ups s_1,..., s_p.

We compare the following methods to solve the (x,u)-formulation (1.2)–(1.4), (1.7) – (1.10) of the MUCP:

- *Default* Cplex:	Default implementation of Cplex used by its C++ API,
- *Callback* Cplex:	Cplex with empty Branch and LazyConstraint Callbacks,
- MOB:	modified orbital branching with no branching rules enforced (Cplex is free to choose the next branching variable),
- SOF:	Static orbitopal fixing for the full-orbitope,
- DOF	Dynamic orbitopal fixing for the full orbitope,
- SOF-S:	Static orbitopal fixing for the full orbitope and sub-orbitope,
- DOF-S:	Dynamic orbitopal fixing for the full orbitope and sub-orbitope.

For methods MOB, SOF, DOF, SOF-S and DOF-S, we also use Cplex C++ API. The fixing (or branching) algorithms are included in Cplex using the so-called Branch Callback, alongside with an empty LazyConstraint Callback to warn Cplex that our methods will remove solutions from the feasible set. Note that such callbacks deactivate some Cplex features designed to improve the efficiency of the overall algorithm. This may induce a bias when comparing results obtained with and without the use of a callback. This is why we compare our methods to Callback Cplex.

Note that in Chapter 3 we only compared our methods to Callback Cplex. Indeed, the presence of the UserCut callback impacted the results instance by instance, but did not seem to incur a loss on average. As opposed to the empty UserCut callback, the LazyConstraint Callback deeply affects the performance of Cplex on the MUCP instances we consider here. This is why we include Default Cplex in our comparison as well.

Note that as shown in Section 5.4, even though MOB+RMRI is slightly better than MOB with no branching rules on UCP instances, we choose to compare our methods to MOB. The rationale behind is that its implementation is straightforward, thus leaving no room to interpretation.

6.4.1 Instances

In order to determine which symmetry-breaking technique performs best with respect to the number of rows and columns of matrix X, we consider various instance sizes (n,T). Namely, we generate instances with $T = 96$ and smaller n : (30, 96), (60,96) and instances with $T = 48$ and larger n: (60, 48), (80,48).

For each pair (n,T), we generate a set of 2-peak-demand MUCP instances with $F = 2,3,4$ as described in Section 1.2.4.

Table 6.1 provides some statistics on the instances characteristics. For each instance, a group is a set of two or more units with same characteristics. Each unit which has not been duplicated is a singleton. The first and second entries column-wise are the number of singletons and groups. The third entry is the mean group size and the fourth entry is the maximum group size. Each

entry row-wise corresponds to the average value obtained over 20 instances with same size (n,T) and same symmetry factor F.

(n,T)	Sym. factor	Nb. singletons	Nb. groups	Group mean size	Group max size
(30,96)	F = 4	1.3	6.5	4.5	6.7
	F = 3	0.4	5.3	6.0	8.7
	F = 2	0.6	4.1	7.6	11.4
(60,96)	F = 4	0.6	7.9	7.8	13.3
	F = 3	0.3	6.0	10.5	16.7
	F = 2	0.2	4.4	14.8	24.9
(60,48)	F = 4	0.8	7.7	7.9	13.1
	F = 3	0.6	5.8	10.9	17.8
	F = 2	0.2	4.8	13.9	23.8
(80,48)	F = 4	0.4	8.0	10.6	18.5
	F = 3	0.5	6.7	12.5	22.2
	F = 2	0.1	4.5	18.9	31.4

Table 6.1: Instance characteristics

Note that the most symmetrical instances are the ones with the highest $\frac{n}{F}$ ratio. Indeed, these instances feature large groups of identical units, and the size of solution orbits grows exponentially with the size of these groups. It is well-known that symmetries dramatically impair the B&B solution process. The highly symmetrical instances are thus expected to be the hardest ones. We also expect that symmetry-breaking techniques will prove useful specifically on these instances.

6.4.2 Static and dynamic orbitopal fixing

The average speed-up achieved by DOF over SOF is given in Table 6.2. The average is computed for each group of 20 instances with same size and symmetry factor.

(30, 96)			(60, 48)		
$F = 4$	$F = 3$	$F = 2$	$F = 4$	$F = 3$	$F = 2$
14.5	3.6	2.6	4.6	11.7	6.7

Table 6.2: Speed-up of DOF with respect to SOF

It is clear that DOF outperforms SOF on each group, by a factor ranging from 2.6 to 14. Thus, we do not consider SOF nor SOF-S in the following experiments. This behavior can be explained as DOF allows for more variable fixings earlier in the B&B tree. Indeed, the orbitopal fixing algorithm propagates a branching decision occurring at r^{th} row (with respect to the lexicographical order) only if there are enough variables already fixed in 1^{st} to $r-1^{th}$ rows. As DOF defines the lexicographical order with respect to the branching decisions, chances are that

many variables are already fixed in each row with rank less than r. Thus, DOF often propagates branching decisions in the B&B tree earlier than SOF does.

Note that MOB also follows the branching decisions, as it branches on a whole variable orbit, *i.e.*, a set of symmetrical variables on a given row. Contrary to DOF, MOB does not account for variables outside the orbit, whereas these variables could be fixed as well.

Instances		Method	#opt	#nodes	#fixings	CPU time
(30,96)	$F = 4$	DC	20	34742	-	34
		CC	12	1669334	-	1506
		MOB	14	794522	49529	1212
		DOF	19	325977	135984	339
		DOF-S	20	96416	74281	129
	$F = 3$	DC	16	823455	-	877
		CC	8	1977613	-	2296
		MOB	12	733875	197964	1578
		DOF	13	831504	667733	1338
		DOF-S	16	484930	564660	899
	$F = 2$	DC	17	367672	-	606
		CC	11	1244729	-	1727
		MOB	12	960300	660193	1525
		DOF	14	575483	698740	1089
		DOF-S	17	496889	736485	1026
(60,96)	$F = 4$	DC	9	1971737	-	1994
		CC	3	1899968	-	3072
		MOB	9	730306	1037813	2082
		DOF	8	932314	3992329	2224
		DOF-S	10	678260	3410927	1828
	$F = 3$	DC	10	1679013	-	2134
		CC	0	1890180	-	3600
		MOB	3	649769	381602	3064
		DOF	5	952878	1813052	2957
		DOF-S	7	633231	2193599	2465
	$F = 2$	DC	9	1669806	-	2128
		CC	0	1295402	-	3600
		MOB	7	562942	281326	2490
		DOF	8	496424	967275	2379
		DOF-S	8	525966	1322964	2199

Table 6.3: Performance indicators relative to the comparison of five methods to solve MUCP instances with symmetries

Instances		Method	#opt	#nodes	#fixings	CPU time
(60,48)	$F = 4$	DC	17	1059290	-	830
		CC	8	2664489	-	2252
		MOB	17	348881	205477	639
		DOF	16	665100	702066	764
		DOF-S	17	431652	694538	558
	$F = 3$	DC	13	1322111	-	1283
		CC	7	2224234	-	2374
		MOB	13	932987	778563	1317
		DOF	15	486352	972444	922
		DOF-S	15	443246	1083904	935
	$F = 2$	DC	17	701617	-	645
		CC	10	1448065	-	1804
		MOB	18	190009	54377	417
		DOF	18	150486	407031	382
		DOF-S	19	135906	449141	325
(80,48)	$F = 4$	DC	8	2423226	-	2168
		CC	1	2653960	-	3420
		MOB	5	1134716	1047231	2798
		DOF	6	1185164	2246156	2607
		DOF-S	9	861262	2476840	2160
	$F = 3$	DC	10	1404892	-	2015
		CC	1	1553426	-	3447
		MOB	2	744775	262750	3247
		DOF	2	936007	1062502	3248
		DOF-S	3	865991	1285128	3169
	$F = 2$	DC	8	2715484	-	2217
		CC	0	3628624	-	3600
		MOB	6	1145092	1150613	2552
		DOF	6	1594025	3597266	2591
		DOF-S	8	1328985	2662087	2269

Table 6.4: Performance indicators relative to the comparison of five methods to solve MUCP instances with symmetries

Instance				$m2$ = Default Cplex		m_2 = Callback Cplex	
(n,T)	Sym	m_1	#opt	opt_Δ	S_{CPU}	opt_Δ	S_{cpu}
(30,96)	$F=4$	MOB	14	-6	0.0902	2	1.57
		DOF	19	-1	0.659	7	11.4
		DOF-S	20	0	0.725	8	12.6
	$F=3$	MOB	12	-4	0.371	4	3.78
		DOF	13	-3	0.507	5	5.17
		DOF-S	16	0	1.05	8	10.7
	$F=2$	MOB	12	-5	0.197	1	2.1
		DOF	14	-3	0.564	3	6
		DOF-S	17	0	0.716	6	7.62
(60,96)	$F=4$	MOB	2	-8	0.218	1	1.36
		DOF	2	-8	0.214	1	1.33
		DOF-S	3	-7	0.218	2	1.36
	$F=3$	MOB	3	-7	0.314	3	2.33
		DOF	5	-5	0.244	5	1.81
		DOF-S	7	-3	0.555	7	4.11
	$F=2$	MOB	7	-2	0.358	7	3.92
		DOF	8	-1	0.493	8	5.39
		DOF-S	8	-1	0.669	8	7.32
(60,48)	$F=4$	MOB	17	0	0.978	9	13.5
		DOF	16	-1	1.45	8	20.1
		DOF-S	17	0	1.92	9	26.5
	$F=3$	MOB	13	0	0.94	6	8.6
		DOF	15	2	2.07	8	18.9
		DOF-S	15	2	2.25	8	20.6
	$F=2$	MOB	18	1	1.84	8	11.7
		DOF	18	1	2.58	8	16.4
		DOF-S	19	2	2.6	9	16.5
(80,48)	$F=4$	MOB	5	-3	0.316	4	2.88
		DOF	6	-2	0.462	5	4.21
		DOF-S	9	1	0.75	8	6.83
	$F=3$	MOB	6	-2	0.637	6	4.97
		DOF	6	-2	0.422	6	3.29
		DOF-S	8	0	0.792	8	6.18
	$F=2$	MOB	9	0	0.701	6	5.22
		DOF	8	-1	0.632	5	4.7
		DOF-S	10	1	1.1	7	8.14

Table 6.5: MOB and dynamic orbitopal fixing (DOF and DOF-S) - average speed-up for various instances compared to Default Cplex and Callback Cplex

6.4.3 Comparison of Cplex, MOB, DOF and DOF-S

We compare five different resolution methods for the MUCP: Default Cplex, Callback Cplex, MOB, DOF and DOF-S. As shown in Table 6.2, dynamic orbitopal fixing outperforms the static variant, thus SOF and SOF-S are not considered.

Tables 6.3 and 6.4 provide, for each method and each group of 20 instances:

#opt:	Number of instances solved to optimality,
#nodes:	Average number of nodes,
#fixings:	Average number of fixings (for MOB, it is the total number of variables fixed during the branching process)
CPU time:	Average CPU time in seconds.

Note that the best feasible solution value is not reported, as all methods are able to find the same best feasible solution value within the time limit.

First note that instances of size (80,48) and, to a lesser extent, of size (60, 96), are the hardest ones: Default Cplex only solves to optimality half of them, and Callback Cplex solves nearly none of them. Further increases in the number n of units or in the number T of time steps would then not be of particular interest, if the corresponding instances are intractable.

Interestingly, increasing the number n of units seems to have more impact on the CPU time than increasing the number T of time steps. Indeed, from instances of size (60,48) to instances of size (80,48), n is only multiplied by a factor 1.3, but the computation time increases by a factor 2. A similar increase in computation time is obtained from instances of size (60,48) to instances of size (60,96), but in this case the number T of time periods has increased by a factor 2. Similarly, from instances of size (30,96) to instances of size (60,48), n increases but T decreases, and both the CPU time and the number of nodes increase. This strong computational impact of parameter n illustrates the polynomiality of the MUCP when n is fixed and T is arbitrary (see Section 2.3).

Note that on average, MOB explores more nodes in comparison with DOF and DOF-S. Even though MOB has more opportunities to fix variables due to the large number of nodes visited, the number of fixings performed by DOF or DOF-S is always much larger (often by at least one order of magnitude). Thus, DOF and DOF-S solve MUCP instances faster, since they branch less thanks to the fixing procedure.

Table 6.5 compares each method m_1, among MOB, DOF and DOF-S, with respect to method m_2, among Default Cplex and Callback Cplex, in terms of average speed-up. The average speed-up is computed on groups of 20 instances of same size (n, T) and same symmetry factor F, as described in Section 6.4.1.

Table 6.5 shows:

(n,T):	Instance size,
F:	Symmetry factor,
m_1:	Method m_1, namely MOB, DOF or DOF-S,
m_2:	Method m_2, namely Default Cplex or Callback Cplex,
#opt:	Number of instances solved to optimality by m_1,
opt_Δ:	Difference in terms of the number of instances solved to optimality by m_1 and by m_2,
S_{CPU}:	Average speed-up by method m_1 with respect to m_2, computed on a group of 20 instances.

In terms of CPU time, MOB, DOF and DOF-S greatly outperform Callback Cplex, but the improvement is larger with DOF and even more significant with DOF-S. Indeed, even on the less symmetrical instances ($(n,T) = (30,96)$ and $F = 4$), MOB outruns Callback Cplex by a factor 1.57 and DOF increases this factor to 11.4. Similarly, on more symmetrical instances $(n,T) = (60,48)$, $F = 4$ (resp. $F = 3$, $F = 2$), MOB outperforms Callback Cplex by a factor 13.5 (resp. 8.6, 11.7) while DOF increases this factor to 20.1 (resp. 18.9, 16.4).

When both symmetries and sub-symmetries are accounted for, the performance is significantly improved. For example, on some of the less symmetrical instances ($(n,T) = (30,96)$ and $F = 3$), DOF outruns Callback Cplex by a factor 5.17 and this factor increases to 10.7 with DOF-S. Similarly, on more symmetrical instances $(n,T) = (60,96)$, $F = 3$ (resp. $F = 2$), DOF outperforms Callback Cplex by a factor 1.81 (resp. 5.39) while DOF-S increases this factor to 4.11 (resp. 7.32). On instances $(n,T) = (60,48)$, $F = 4$, DOF-S is even faster than Callback Cplex by a factor 26.5.

As observed in [70], there is a huge performance gap between Callback Cplex and Default Cplex. Thus, even if MOB, DOF and DOF-S substantially outperforms Callback Cplex in each instance group, it is sometimes not enough to close the performance gap between Default and Callback Cplex, especially for instances with small n. On the opposite, for large n instances where symmetries are a major source of difficulty, DOF and DOF-S clearly outperforms Default Cplex.

Typically, when T is large compared to n ($i.e.$, on instances of size (60,96) and (30,96)) it seems that non symmetry-related difficulties arise, and none of the compared methods catch up with Default Cplex. In this context, the cost of applying symmetry-breaking techniques (including the performance loss induced by the use of a Callback) seems too important compared to the impact of symmetries. The performance loss is less important with DOF and DOF-S than it is with MOB. DOF-S is the method that is the closest to catch up with Default Cplex. Indeed, for $(n,T) = (30,96)$ instances, it solves to optimality as many instances as Default Cplex, and on $F = 3$ instances of size (30,96) DOF-S even slightly improves Default Cplex, while MOB is slower than Default Cplex by a factor 3.

On the opposite, when n is large compared to T ($i.e.$, on instances of size (80,48) and (60,48)),

symmetry seems to be a major factor of computational difficulty. Indeed, DOF-S performs quite well in this context and solves to optimality some instances Default Cplex cannot. For example, on instances $(n,T) = (60,48)$, $F = 2$ (resp. $F = 3$), DOF-S solves two more instances to optimality than Default Cplex. DOF and MOB do not perform as well as DOF-S in this respect. On instances of size (60,48), DOF and DOF-S outrun Default Cplex by a factor 2, while MOB is closer to a factor 1. When n increases to 80, DOF-S achieves a speed-up of 1.1 compared to Default Cplex on the most symmetrical instances ($F = 2$), while MOB and DOF stay behind with a speed-up around 0.7 relatively to Default Cplex. Moreover, DOF-S solves more instances to optimality than Default Cplex. For less symmetrical instances with $n = 80$, *i.e.*, $F = 3$ and $F = 4$ groups, none of the compared methods are able to outrun Default Cplex in terms of CPU time. It seems that non-symmetry related difficulties inherent to the MUCP arise in these instances featuring a large number of distinct units. In this context, DOF-S is the method closest to catch up with Default Cplex. Indeed, on both groups of instances, the speed-up provided by DOF is around 0.8, whereas this factor ranges from 0.3 to 0.6 for MOB and DOF. While Callback Cplex solves to optimality only one instance out of forty, DOF-S proves its efficiency by solving even more instances to optimality than Default Cplex.

Concluding remarks and perspectives

We define a linear time orbitopal fixing algorithm for the full orbitope. This algorithm is optimal, in the sense that at any node a in the B&B tree, any variable that can be fixed, with respect to the lexicographical order, is fixed by the algorithm. We propose a dynamic version of the orbitopal fixing algorithm, where the lexicographical order at node a is defined with respect to the branching decisions leading to a. We show that the proposed orbitopal fixing algorithm can be also applied to handle sub-symmetries related to sub-orbitopes.

For MUCP instances, experimental results show that the dynamic variant of our algorithm performs much better than the static variant. Moreover, it is clear that sub-symmetries greatly impair the solution process for MUCP instances, since dynamic orbitopal fixing for both full orbitope and full sub-orbitope (DOF-S) performs even better than dynamic orbitopal fixing for the full orbitope (DOF). Finally, our experiments show that our approach is competitive with commercial solvers like Cplex and state-of-the-art techniques like modified orbital branching (MOB). Even if MOB already improves Callback Cplex, the improvement is even more significant with our methods DOF and DOF-S. Furthermore, even if there is a huge performance gap between Callback Cplex and Default Cplex, DOF-S is able to outrun Default Cplex by a factor 2 on some of the most symmetrical instances.

An option to improve the efficiency of the orbitopal fixing algorithm, implemented within Cplex's framework, would be to store the variables already fixed and their bounds in our own data structure, in order to avoid the costly calls to Cplex's getters at each node.

As a perspective, it would be interesting to extend orbitopal fixing to full orbitopes under other group actions, for example the cyclic group. Another approach to handle symmetries related to the symmetric or the cyclic group would be to find a new set of representatives whose convex hull would be easier to describe than the full orbitope.

Finally, there is a wide range of problems featuring all column permutation symmetries and sub-symmetries, in particular many variants of the UCP, on which our approach could be applied. Other examples of such problems can be found among covering problems, whose solution matrix has at least one 1-entry per row, like bin-packing variants. Even though computing the exact fixing has been shown NP-hard in this case, our orbitopal fixing algorithm, designed for full orbitopes, can be used to compute valid variable fixings in a covering orbitope as well. In this case, there is no guarantee that fixings are done as early as possible in the tree, because the special structure of covering orbitopes may induce possible fixings that would not be correct in a full orbitope. Nevertheless, this fixing algorithm breaks all column-permutation related symmetries at some point in the B&B tree, which may be sufficient to overcome the computational difficulties arising from the highly symmetrical nature of these problems.

CHAPTER 7

SUB-SYMMETRY BREAKING INEQUALITIES

We propose a general framework to build full symmetry-breaking inequalities in order to handle sub-symmetries arising from solution subsets whose symmetry groups contain the symmetric group acting on some sub-columns. One additional variable per subset Q considered may be needed in these inequalities, depending whether variables x are sufficient to indicate that "x belongs to subset Q". The proposed framework is applied to derive such inequalities when the symmetry group is the symmetric group $\mathfrak{S}_n$ acting on the columns. It is also applied to derive inequalities breaking both symmetries and sub-symmetries in the MUCP. We present experimental results comparing these sub-symmetry-breaking inequalities to state-of-the-art symmetry-breaking formulations, such as the MUCP formulation featuring inequality (5.6) [55] (see Section 5.3.1), aggregated (x,u) or aggregated interval MUCP formulations [50] (see Section 5.4). When the MUCP is considered, the integer decomposition property holds for the (x,u) formulation and thus efficient aggregation techniques apply [50]. When the ramp-constrained MUCP is considered, the integer decomposition property (see Theorem 1.3) does not hold anymore for the (x,u) MUCP formulation, then the corresponding aggregated solutions can no longer be disaggregated. We show that our inequalities outperform all above mentioned formulations in the ramp-constrained case.

For a given solution subset Q, the symmetry group $\mathscr{G}_Q$ of the corresponding subproblem is different from $\mathscr{G}$ and may contain symmetries undetected in $\mathscr{G}$. In practice it is too expansive to compute the symmetry group for every subset $Q \subset \mathscr{X}$. However for many problems, symmetries of $\mathscr{G}$ can be deduced from the problem's structure, and so can symmetries of $\mathscr{G}_Q$, for some particular solution subsets Q. In this case, symmetries of $\mathscr{G}_Q$ are a priori known, and thus do not need to be computed. Such symmetries may be handled together with symmetries of $\mathscr{G}$. In this section, we introduce sub-symmetry-breaking inequalities designed to simultaneously handle symmetries

and sub-symmetries in symmetric groups.

In Sections 7.1 and 7.2, we describe the framework. In Section 7.3, an application to the symmetric group case is presented. In Section 7.4, the framework is applied to derive sub-symmetry-breaking inequalities dedicated to the MUCP. Experimental results on MUCP instances (with or without ramp constraints) are presented in Section 7.5.

The results presented in this chapter have been published in [11].

7.1 Definition and validity

Consider a set $\mathbb{S}$ of solution subsets Q_s, $s \in \{1,...,q\}$, such that each subset Q_s, $s \in \{1,...,q\}$, is sub-symmetric with respect to (R_s, C_s). Consider integer variable z_s, $s \in \{1,...,q\}$, such that $z_s = 0$ if variable $x \in Q_s$, and such that $z_s \geq 1$ if $x \notin Q_s$. For any $x \in \mathscr{X}$, function Z associates x to a vector $Z(x)$ such that z_s, $s \in \{1,...,q\}$, is the s^{th} component of $Z(x)$ denoted by $Z_s(x)$

Note that in many cases, function Z is linear, *i.e.*, each integer variable z_s is a linear expression of variables x. In such cases, no additional variable z_s is needed. In some cases where function Z is not linear, variable z_s can be linearly expressed from variables x using only a few additional inequalities or integer variables.

Given c, $c' \in C_s$ such that $c < c'$, the *sub-symmetry-breaking inequality*, denoted by $(Q_s(c,c'))$, is defined as follows.

$$x_{r_1,c'} \leq z_s + x_{r_1,c} \qquad \text{where } r_1 = \min(R_s) \tag{7.1}$$

For each orbit O_k^s, $k \in \{1,...,o_s\}$, of $\mathscr{G}_{Q_s}$, $s \in \{1,...,q\}$, the chosen representative is the lex-max of orbit O_k^s with respect to (R_s, C_s). Then by Property 5.3, this set of representatives is orbit-compatible. In particular, solution set $\mathscr{X}$ can be restricted to the set of representatives by considering its intersection with the full sub-orbitope $\mathscr{P}_{sub}(\mathbb{S})$. If $x \in Q_s$, inequality $(Q_s(c,c'))$ enforces that the first row of submatrix $x(R_s, C_s)$ is lexicographically non-increasing, hence the following result.

Lemma 7.1 (Validity)**.** *If $x \in \mathscr{P}_{sub}(\mathbb{S})$, then $(x, Z(x))$ satisfies inequality $(Q_s(c,c'))$ for each $s \in \{1,...,q\}$ and c, $c' \in C_s$ such that $c < c'$.*

Note that an inequality similar to (7.1) applied to a row of R_s distinct from r_1 may not be valid when used alongside with (7.1), as shown in Example 7.1.

Example 7.1. *Let $\mathbb{S} = \{Q_1\}$, $q = 1$, where subset Q_1 is as follows.*

$$Q_1 = \left\{ x \in \mathscr{P}(4,3) \cap \mathscr{X} \mid \sum_{c=1}^{3} x_{2,c} = 3 \right\}$$

Let us suppose the symmetry group of Q_1 is the sub-symmetric group with respect to submatrix $(\{3,4\},\{1,2,3\})$. Variable z_1 can be defined using equality $z_1 = 3 - \sum_{c=1}^{3} x_{2,c}$. Note that $z_1 = Z_1(x) = 0$

when $\sum_{c=1}^{3} x_{2,c} = 3$*, i.e.,* $x \in Q_1$*, and is positive otherwise. Here the first row in* R_1 *is* $r_1 = \min(R_1) = 3$*, thus given* c, $c' \in \{1,2,3\}$, $c < c'$*, inequality* $(Q_1(c,c'))$ *is as follows*

$$x_{3,c'} \leq \left(3 - \sum_{j=1}^{3} x_{2,j}\right) + x_{3,c} \tag{7.2}$$

Inequality $(Q_1(c,c'))$ *enforces that row 3 of a solution matrix x is lexicographically ordered, i.e.,* $x_{3,1} \geq x_{3,2} \geq x_{3,3}$*, whenever* $\sum_{c=1}^{3} x_{2,c} = 3$.

Now consider solutions x^1, $x^2 \in Q_1$*:*

$$x^1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \qquad \textit{and} \qquad x^2 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Inequality (7.2) *cuts off solution* x^2 *from the feasible set. Inequality* (7.3) *corresponds to inequality* (7.1) *applied to row 4:*

$$x_{4,c'} \leq \left(3 - \sum_{j=1}^{3} x_{2,j}\right) + x_{4,c} \tag{7.3}$$

Inequality (7.3) *would cut off* x^1*. This shows that inequalities* (7.2) *and* (7.3) *cannot be used simultaneously.*

Note that in the general case, inequalities (7.1) may only be partial symmetry-breaking. Indeed, for given $s \in \{1,...,q\}$ and c, $c' \in C_s$ such that $c < c'$, inequality $(Q_s(c,c'))$ only enforces that the first row of submatrix $x(R_s, C_s)$ is lexicographically non-increasing when $x \in Q_s$. In the case when $x_{r_1,c'} < x_{r_1,c}$, then sub-columns $x(R_s,\{c'\}) \prec x(R_s,\{c\})$. Otherwise, when $x_{r_1,c'} = x_{r_1,c}$, inequality (7.1) is not sufficient to select the lexmax representatives.

To enforce a lexicographical order, subsequent rows of submatrix $x(R_s, C_s)$ should be considered until a tie-break row is found. It is shown in the next section that inequalities $(Q_s(c,c'))$ for all $s \in \{1,...,q\}$ and $c < c' \in C_s$ enforce that $x \in \mathscr{P}_{sub}(\mathbb{S})$ provided a tie-break condition on set $\mathbb{S}$ is fulfilled.

7.2 Full symmetry-breaking sufficient condition

In this section, we introduce a condition for inequalities (7.1) to be full symmetry-breaking.

For each $s \in \{1,...,q\}$, consider $R_s = \{r_1^s,...,r_{|R_s|}^s\}$ and $C_s = \{c_1^s,...,c_{|C_s|}^s\}$, where $r_1^s < ... < r_{|R_s|}^s$ and $c_1^s < ... < c_{|C_s|}^s$. For given $s \in \{1,...,q\}$ and any two columns $c_{l-1}^s, c_l^s \in C_s$, if there is a solution $x \in Q_s$ such that columns c_{l-1}^s and c_l^s are equal from row r_1^s to row r_{k-1}^s, it must be ensured that row r_k^s is lexicographically non increasing, *i.e.*, $x_{r_k^s, c_{l-1}^s} \leq x_{r_k^s, c_l^s}$. The key idea is to exhibit another set $Q_p \in \mathbb{S}$ for quartet (Q_s, k, l, x), such that Q_p contains x and is sub-symmetric with respect to

(R_p, C_p), where the first row of R_p is r^s_k and C_p contains columns c^s_{l-1} and c^s_l. Then inequality $(Q_p(c^s_{l-1}, c^s_l))$ will ensure that $x_{r^s_k, c^s_{l-1}} \geq x_{r^s_k, c^s_l}$. For each quartet (Q_s, k, l, x), the existence of such a subset Q_p in $\mathbb{S}$ will be ensured by tie-break condition $(\mathscr{C})$, defined as follows:

$$(\mathscr{C}) \quad \begin{cases} \forall s \in \{1,...,q\},\ \forall k \in \{2,...,|R_s|\},\ \forall l \in \{2,...,|C_s|\} \\ \text{If } x \in Q_s \text{ such that } x_{r^s_{k'}, c^s_{l-1}} = x_{r^s_{k'}, c^s_l},\ \forall k' \in \{1,...,k-1\}, \\ \text{then there exists } p \in \{1,...,q\} \text{ such that } x \in Q_p,\ C_p \supseteq \{c^s_{l-1}, c^s_l\} \text{ and } r^s_k = \min(R_p) \end{cases}$$

If condition $(\mathscr{C})$ holds, inequalities $(Q_s(c^s_{l-1}, c^s_l))$, $\forall s \in \{1,...,q\}$, $\forall l \in \{2,...,|C_s|\}$ exactly restrict the solution set to the representative set $\mathscr{X} \cap \mathscr{P}_{sub}(\mathbb{S})$. They are therefore full symmetry breaking, with respect to the sub-symmetries defined by $\mathbb{S}$. This gives the idea of the proof for the following theorem.

Theorem 7.1. *If condition $(\mathscr{C})$ holds, then:*

(i) $(x, Z(x))$ *satisfies* $(Q_s(c^s_{l-1}, c^s_l))$, $\forall s \in \{1,...,q\}$, $\forall l \in \{2,...,|C_s|\}$

(ii) $x \in \mathscr{P}_{sub}(\mathbb{S})$

are equivalent.

For general set $\mathbb{S}$, condition $(\mathscr{C})$ may not hold. Fortunately, it will be shown that we can construct from $\mathbb{S}$ another set $\widetilde{\mathbb{S}}$ satisfying $(\mathscr{C})$ and such that $\mathscr{P}_{sub}(\widetilde{\mathbb{S}}) = \mathscr{P}_{sub}(\mathbb{S})$.

The idea is to divide each Q_s, $s \in \{1,...,q\}$ in smaller subsets such that for each row $r^s_k \in R_s$ and each column $c^s_l \in C_s$, there is a subset Q, which is sub-symmetric with respect to $(R, C) = (\{r^s_k, ..., r^s_{|R_s|}\}, \{c^s_{l-1}, c^s_l\})$.

Set $\widetilde{\mathbb{S}}$ is defined as follows.

$$\widetilde{\mathbb{S}} = \left\{ \widetilde{Q}_s(k,l) \mid s \in \{1,...,q\},\ k \in \{1,...,|R_s|\},\ l \in \{2,...,|C_s|\} \right\}$$

where for each $s \in \{1,...,q\}$, for each $l \in \{2,...,|C_s|\}$, for each $k \in \{1,...,|R_s|\}$,

$$\widetilde{Q}_s(k,l) = \left\{ x \in Q_s \mid x_{r, c^s_{l-1}} = x_{r, c^s_l},\ \forall r \in \{r^s_1, ..., r^s_{k-1}\} \right\}$$

Note that for solution $x \in Q_s$ such that columns c^s_{l-1} and c^s_l are equal from row r^s_1 to row r^s_{k-1}, the set exhibited for quartet (Q_s, k, l, x) is $\widetilde{Q}_s(k,l)$. Note also that $\widetilde{Q}_s(1,l) = Q_s$, $l \in \{2,...,|C_s|\}$.

We thus have the following result:

Lemma 7.2. *Set $\widetilde{\mathbb{S}}$ satisfies $(\mathscr{C})$ and is such that $\mathscr{P}_{sub}(\widetilde{\mathbb{S}}) = \mathscr{P}_{sub}(\mathbb{S})$.*

Proof. The symmetry group of $\widetilde{Q}_s(k,l)$ is the sub-symmetric group with respect to $(R, C) = (\{r^s_k, ..., r^s_{|R_s|}\}, \{c^s_{l-1}, c^s_l\})$. Thus if some solution $x \in Q_s$ is such that columns c^s_{l-1} and c^s_l are equal from row r^s_1 to row r^s_{k-1}, then subset $\widetilde{Q}_s(k,l)$ contains x and is such that $C \supseteq \{c^s_{l-1}, c^s_l\}$ and $\min(R) = r^s_k$. Condition $(\mathscr{C})$ is therefore satisfied by $\widetilde{\mathbb{S}}$. It can be readily checked that the full sub-orbitopes defined by $\widetilde{\mathbb{S}}$ and $\mathbb{S}$ are the same. ■

It follows, from Theorem 7.1, that inequalities $(Q(c,c'))$, $c < c' \in C$, $Q \in \widetilde{\mathbb{S}}$ are full symmetry-breaking with respect to the sub-symmetries defined by $\mathbb{S}$.

Corollary 7.1. *If for each $Q \in \widetilde{\mathbb{S}}$, $(x, Z(x))$ satisfies inequality $(Q(c,c'))$, $\forall c < c' \in C$, then $x \in \mathscr{P}_{sub}(\mathbb{S})$.*

We can then consider $\widetilde{\mathbb{S}}$ instead of $\mathbb{S}$. This implies to add one inequality at least (resp. one variable at most), per subset $Q \in \widetilde{\mathbb{S}}$, *i.e.*, $O(qmn)$ inequalities at least (resp. variables at most).

Example 7.2. *Referring to Example 7.1, $\widetilde{\mathbb{S}} = \{\widetilde{Q}_1(1,l), \widetilde{Q}_1(2,l),\ l \in \{2,3\}\}$. For each $l \in \{2,3\}$, $\widetilde{Q}_1(1,l) = Q_1$ as for any s, $\widetilde{Q}_s(k,l) = Q_s$ whenever $k = 1$. We also have $\widetilde{Q}_1(2,l) = \{x \in Q_1 \mid x_{3,l-1} = x_{3,l}\}$. For each $l \in \{2,3\}$, $\widetilde{z}_l$ associated to subset $\widetilde{Q}_1(2,l)$ can be expressed as follows: $\widetilde{z}_l = 2z_1 + (x_{3,l-1} - x_{3,l})$. Indeed, when $z_1 = 0$, inequality (7.2) becomes $x_{3,l-1} \leq x_{3,l}$. Thus, $\widetilde{z}_l = 0$ if $x_{3,l-1} = x_{3,l}$ and $z_l \geq 1$ otherwise. When $z_1 = 1$, $\widetilde{z}_l \geq 1$. Hence the following inequalities are full symmetry-breaking:*

$$x_{3,l-1} \leq \left(3 - \textstyle\sum_{j=1}^{3} x_{2,j}\right) + x_{3,l} \qquad \forall l \in \{2,3\}$$
$$x_{4,l-1} \leq \left(6 + x_{3,l-1} - x_{3,l} - 2\textstyle\sum_{j=1}^{3} x_{2,j}\right) + x_{4,l} \qquad \forall l \in \{2,3\}$$

In Sections 7.3 and 7.4, inequalities (7.1) are built in a more straightforward way, in the sense that set $\mathbb{S}$ already satisfies condition (C) in the two applications studied.

7.3 Application to the symmetric group case

In this section, we apply the framework of Sections 7.1 and 7.2 to any problem whose symmetry group $\mathscr{G}$ is the symmetric group $\mathfrak{S}_n$ acting on the columns. The collection $\mathbb{S}_{\mathfrak{S}}$ of subsets considered will lead to inequalities restricting any solution $x \in \mathscr{X}$ to be in the full orbitope. These inequalities feature variables z which can be explicitly expressed from x with $O(mn)$ linear inequalities. Here, the sub-symmetries considered are restrictions of symmetries' actions to solution subsets.

A complete linear description of the 2-column full orbitope, featuring additional integer variables, is proposed in [56]. In the general n-column case, we show that these inequalities can also be derived using the framework described in Sections 7.1 and 7.2, and can be used as full symmetry-breaking inequalities.

We consider $\mathbb{S}_{\mathfrak{S}} = \{Q_{i,j},\ i \in \{0\} \cup \{1,...,m-1\},\ j \in \{2,...,n\}\}$, where

$$Q_{i,j} = \left\{x \in \mathscr{X} \mid x_{i',j-1} = x_{i',j}\ \forall i' \in \{1,...,i\}\right\}.$$

Subset $Q_{i,j}$ is the set of feasible solutions such that columns $j-1$ and j are equal from row 1 to row i. Note that $Q_{0,j} = \mathscr{X}$. The symmetry group of $Q_{i,j}$ is then the sub-symmetric group with respect to $(R_i, \{j-1, j\})$ where $R_i = \{i+1,...,m\}$. It can be readily checked that in this case, $\mathbb{S}$ already satisfies condition $(\mathscr{C})$.

Let variable $z_{i,j}$ be such that $z_{i,j} = 0$ if $x \in Q_{i,j}$ and 1 otherwise. Note that for all $j \in \{2,...,n\}$, $Q_{0,j} = \mathscr{X}$, thus $z_{0,j} = 0$, $\forall x \in \mathscr{X}$. Note also that $\mathscr{X} \cap \mathscr{P}_{sub}(\mathbb{S}_{\mathfrak{S}})$ is a subset of the full orbitope.

Thus, given that the columns of any $x \in \mathscr{X} \cap \mathscr{P}_{sub}(\mathbb{S}_{\mathfrak{S}})$ are in a non-increasing lexicographical order, function Z is such that $Z(x) = z$, where z satisfies the following linear inequalities.

$$\begin{cases} z_{1,j-1} = x_{1,j-1} - x_{1,j} & \forall j \in \{2,...,n\} & \text{(7.4a)} \\ z_{i,j-1} \leq z_{i-1,j-1} + x_{i,j-1} & \forall i \in \{2,...,m\},\ j \in \{2,...,n\} & \text{(7.4b)} \\ z_{i,j-1} + x_{i,j} \leq 1 + z_{i-1,j-1} & \forall i \in \{2,...,m\},\ j \in \{2,...,n\} & \text{(7.4c)} \\ x_{i,j-1} \leq z_{i,j-1} + x_{i,j} & \forall i \in \{2,...,m\},\ j \in \{2,...,n\} & \text{(7.4d)} \\ z_{i-1,j-1} \leq z_{i,j-1} & \forall i \in \{2,...,m\},\ j \in \{2,...,n\} & \text{(7.4e)} \end{cases}$$

Constraint (7.4a) sets variable $z_{1,j-1}$ to 1 whenever columns $j-1$ and j are different and in a non-increasing lexicographical order on row 1, and to 0 when they are equal. Constraints (7.4b) and (7.4c) set variable $z_{i,j-1}$ to 0 when $z_{i-1,j-1} = 0$ and columns $j-1$ and j are equal on row i. Constraint (7.4d) sets variable $z_{i,j-1}$ to 1 if columns $j-1$ and j are different and in a non-increasing lexicographical order on row i. Constraint (7.4e) sets $z_{i,j-1}$ to 1 when variable $z_{i-1,j-1} = 1$, *i.e.*, when columns $j-1$ and j are different before row i.

For each $i \in \{0,...,m-1\}$ and $j \in \{2,...,n\}$ inequality (7.1) is inequality $(Q_{i,j}(j-1,j))$ as follows:

$$x_{i+1,j} \leq z_{i,j-1} + x_{i+1,j-1} \qquad \forall i \in \{1,...,m\},\ \forall j \in \{2,...,n\}$$

It ensures that if columns $j-1$ and j of x are equal from row 1 to i, then row $i+1$ is in a non-increasing lexicographical order.

Note that if $z_{i-1,j-1} - z_{i,j-1} = -1$ then necessarily $x_{i,j} = 0$. Thus inequality $((Q_{i,j}(j-1,j)))$ can be lifted to

$$x_{i,j} \leq (2z_{i-1,j-1} - z_{i,j-1}) + x_{i,j-1} \tag{7.5}$$

In the special case when $n = 2$, by replacing variable $z_{i,j}$ by $y_{i,j}$ where $z_{i,j} = 1 - \sum_{i'=1}^{i} y_{i,j}$, for each $i \in \{1,...,m\}$, $j \in \{1,2\}$, inequalities (7.4a)–(7.5) yield the complete linear description of the 2-column full orbitope proposed in [56].

In the general n-column case, inequalities (7.4a)-(7.5) are still full symmetry-breaking (by Theorem 7.1), and then can be used in practice to restrict the feasible set to any full orbitope. In this case, $O(mn)$ additional variables and constraints are needed.

7.4 Application to the Unit Commitment Problem

The framework of Sections 7.1 and 7.2 is now applied to the MUCP, which features many sub-symmetries non detected by the symmetry group $\mathscr{G}$.

7.4.1 Sub-symmetry-breaking inequalities for the MUCP

For each time period $t \in \{1,...,\mathcal{T}\}$ and any two consecutive units j_k^h, j_{k+1}^h of type h, $k \in \{1,...,n_h-1\}$, consider the following subsets of $\mathscr{X}_{MUCP}$:

$$\overline{Q}_{k,h}^t = \left\{x \in \mathscr{X}_{MUCP} \mid x_j^{t'} = 0,\ \forall t' \in \{t-\ell^h,...,t-1\},\ \forall j \in \{j_k^h, j_{k+1}^h\}\right\}$$

$$\underline{Q}_{k,h}^t = \left\{x \in \mathscr{X}_{MUCP} \mid x_j^{t'} = 1,\ \forall t' \in \{t-L^h,...,t-1\},\ \forall j \in \{j_k^h, j_{k+1}^h\}\right\}$$

where ℓ^h (resp. L^h) is the minimum down (resp. up) time of units of type h.

Note that $\overline{Q}_{k,h}^t$ and $\underline{Q}_{k,h}^t$ are different from subsets $Q_{i,j}$ defined in Section 7.3. Actually, $Q_{t,j_{k+1}^h} \subset \overline{Q}_{k,h}^t$ and $Q_{t,j_{k+1}^h} \subset \underline{Q}_{k,h}^t$.

Let $\mathscr{G}_{\overline{Q}_{k,h}^t}$ and $\mathscr{G}_{\underline{Q}_{k,h}^t}$ be the sub-symmetry groups associated to $\overline{Q}_{k,h}^t$ and $\underline{Q}_{k,h}^t$, $t \in \{1,...,\mathcal{T}\}$, $h \in \{1,...,H\}$, $k \in \{1,...,n_h-1\}$. The sub-symmetries in $\mathscr{G}_{\overline{Q}_{k,h}^t}$ (resp. $\mathscr{G}_{\underline{Q}_{k,h}^t}$) are called *start-up sub-symmetries* (resp. *shut-down sub-symmetries*). Most of these sub-symmetries are not detected in the symmetry group of the MUCP.

Groups $\mathscr{G}_{\overline{Q}_{k,h}^t}$ and $\mathscr{G}_{\underline{Q}_{k,h}^t}$ contain the sub-symmetric group associated to the submatrix defined by rows and columns $(\{t,...,T\},\{j_k^h, j_{k+1}^h\})$.

Applying results from Section 7.1, variables $\overline{z}_{k,h}^t$ and $\underline{z}_{k,h}^t$, indicating whether $x \in \overline{Q}_{k,h}^t$ and $x \in \underline{Q}_{k,h}^t$ respectively, can be directly derived from variables x and u:

$$\overline{z}_{k,h}^t = x_{t-\ell^h}^{j'} + \sum_{t'=t-\ell^h+1}^{t-1} u_{t'}^{j'} + x_{t-\ell^h}^{j} + \sum_{t'=t-\ell^h+1}^{t-1} u_{t'}^{j}$$

$$\underline{z}_{k,h}^t = 1 - x_{t-L^h}^{j'} + \sum_{t'=t-L^h+1}^{t-1} w_{t'}^{j'} + 1 - x_{t-L^h}^{j} + \sum_{t'=t-L^h+1}^{t-1} w_{t'}^{j}$$

where $j = j_k^h$ and $j' = j_{k+1}^h$ for sake of clarity.

Consider $\mathbb{S}_{MUCP} = \left\{\overline{Q}_{k,h}^t,\ \underline{Q}_{k,h}^t,\ t \in \{1,...,\mathcal{T}\},\ h \in \{1,...,H\},\ k \in \{1,...,n_h-1\}\right\}$. In this case, set $\mathbb{S}$ directly satisfies condition $\mathscr{C}$.

For each $h \in \{1,...,H\}$, $k \in \{1,...,n_h-1\}$ and $t \in \{1,...,\mathcal{T}\}$, inequalities $(\overline{Q}_{k,h}^t(j,j'))$ and $(\underline{Q}_{k,h}^t(j,j'))$, where $j = j_k^h$ and $j' = j_{k+1}^h$, are as follows.

$$x_t^{j'} \leq \left[x_{t-\ell^h}^{j'} + \sum_{t'=t-\ell^h+1}^{t-1} u_{t'}^{j'}\right] + \left[x_{t-\ell^h}^{j} + \sum_{t'=t-\ell^h+1}^{t-1} u_{t'}^{j}\right] + x_t^j$$

$$x_t^{j'} \leq \left[1 - x_{t-L^h}^{j'} + \sum_{t'=t-L^h+1}^{t-1} w_{t'}^{j'}\right] + \left[1 - x_{t-L^h}^{j} + \sum_{t'=t-L^h+1}^{t-1} w_{t'}^{j}\right] + x_t^j$$

Strengthening symmetry-breaking inequalities Inequalities $(\overline{Q}_{k,h}^t(j,j'))$ and $(\underline{Q}_{k,h}^t(j,j'))$ can be further strengthened, using the relationship between variables x and u.

First note that by definition of variables w:

$$x_t^{j'} - \left[x_{t-\ell^h}^{j'} + \sum_{t'=t-\ell^h+1}^{t-1} u_{t'}^{j'}\right] = u_t^{j'} - \sum_{t'=t-\ell^h+1}^{t} w_{t'}^{j'}$$

$$x_t^{j} + \left[1 - x_{t-L^h}^{j} + \sum_{t'=t-L^h+1}^{t-1} w_{t'}^{j}\right] = -w_t^{j} + 1 + \sum_{t'=t-L^h+1}^{t} u_{t'}^{j}$$

As if $u_t^{j'} = 1$ (resp. $w_t^j = 1$), then $\sum_{t'=t-\ell^h+1}^{t} w_{t'}^{j'} = 0$ (resp. $\sum_{t'=t-L^h+1}^{t} u_{t'}^{j} = 0$), the following *Start-Up-Ready* and *Shut-Down-Ready* inequalities are valid and stronger than inequalities $(\overline{Q}_{k,h}^t(j,j'))$ and $(\underline{Q}_{k,h}^t(j,j'))$.

$$u_t^{j'} \leq \left[x_{t-\ell^h}^{j} + \sum_{t'=t-\ell^h+1}^{t-1} u_{t'}^{j}\right] + x_t^j \tag{7.6}$$

$$w_t^{j} \leq \left[1 - x_{t-L^h}^{j'} + \sum_{t'=t-L^h+1}^{t-1} w_{t'}^{j'}\right] + 1 - x_t^{j'} \tag{7.7}$$

Note that for any $h \in \{1,...,H\}$ and $k \in \{1,...,n_h - 1\}$, $\overline{Q}_{k,h}^1 = \underline{Q}_{k,h}^1 = \mathscr{X}_{MUCP}$. As condition (C) is satisfied by $\mathbb{S}_{MUCP}$, any $x = (x_1,...,x_H)$ satisfying inequalities (7.6) and (7.7) is such that x_h is in the $T \times n_h$ full orbitope, $h \in \{1,...,H\}$. Hence inequalities (7.6) and (7.7) ensure in particular that any solution x_h is in the full orbitope.

7.4.2 Sub-symmetry-breaking inequalities for the ramp-constrained MUCP

The MUCP formulation including ramp constraints can be further strengthened with valid inequalities as proposed in [71, 75]. As the aim of this chapter is to compare symmetry-breaking techniques, we will only consider the classical MUCP formulation (1.2)–(1.4), (1.7) – (1.10) with ramp-constraints (1.11) – (1.12), as done in [49, 70].

When ramp-constraints are considered, the symmetry group of set $\overline{Q}_{k,h}^t$ still contains the sub-symmetric group associated to the submatrix defined by rows and columns $(\{t,...,T\},\{j_k^h, j_{k+1}^h\})$. Therefore, inequalities (7.6) can still be used.

However the symmetry group of set $\underline{Q}_{k,h}^t$ no longer contains the sub-symmetric group associated to the submatrix defined by rows and columns $(\{t,...,T\},\{j_k^h, j_{k+1}^h\})$. Indeed, if two identical units have been up for at least L^h time periods at time $t-1$, they may produce distinct power values at time $t-1$ and thus, because of ramp constraints, their up/down trajectories from time t to T cannot be permuted. Therefore, inequalities (7.7) can no longer be used.

Note that when two identical ramp-constrained units are ready to shut down, there still exist some sub-symmetries that could be exploited. These sub-symmetries are more intricate because they depend, for example, on the quantity of power produced by both units, or on the time of their last start-up.

7.5 Experimental results

In this section, we compare various formulations for the MUCP with or without ramp constraints. Some symmetry-breaking techniques need to interfere with the branching process. These are typically implemented using a callback instruction which deeply affects the performance of commercial solvers like Cplex. Consequently in our computational comparison, we only consider symmetry-breaking techniques that do not require the use of a callback.

7.5.1 Experimental settings

In this section, we compare various symmetry-breaking formulations for the MUCP with or without ramp-constraints.

As shown in [70], neither Friedman inequalities (5.4) nor column inequalities (5.5) are competitive with respect to the classical UCP formulation when solved by Cplex.

On the opposite, the weaker form of Friedman inequality (5.6) has been shown in [55] to outperform Default Cplex.

In [50], the authors propose to break symmetries of the UCP by aggregating variables corresponding to identical units.

Hence the following formulations for the MUCP are compared:

- $F(x,u)$: (x,u)-formulation (1.2)–(1.4), (1.7) – (1.10)
- A-$(\widetilde{x},\widetilde{u})$: Aggregated $(\widetilde{x},\widetilde{u})$-formulation (5.7)–(5.12) (only when disaggregation applies)
- Int$(\widetilde{y})$: Aggregated interval formulation (5.18)–(5.21)
- $W(x,u)$: (x,u)-formulation (1.2)–(1.4), (1.7) – (1.10) with weaker Friedman inequalities (5.6)
- $F(x,u,z)$: (x,u)-formulation (1.2)–(1.4), (1.7) – (1.10) with variables z, inequalities (7.4a)–(7.4e) and sub-symmetry-breaking inequalities (7.5)
- $LF(x,u)$: (x,u)-formulation (1.2)–(1.4), (1.7) – (1.10) with sub-symmetry-breaking inequalities (7.6)-(7.7).

Formulation $F(x,u,z)$ is obtained from the classical MUCP formulation $F(x,u)$ by a direct use of the inequalities given in Section 7.3. As seen in Section 7.4, taking into account sub-symmetries in the MUCP leads to formulation $LF(x,u)$ featuring lifted symmetry breaking-inequalities (7.6) and (7.7), namely Start-up-ready and Shut-down-ready inequalities, in place of inequalities (7.4a)–(7.5). Note that the start-up and shut-down sub-symmetries of the MUCP are not handled by formulations $F(x,u)$, $W(x,u)$ and $F(x,u,z)$.

Formulations $F(x,u)$, $W(x,u)$, $F(x,u,z)$ and $LF(x,u)$ feature $O(nT)$ variables while formulation A-$(\widetilde{x},\widetilde{u})$ (resp. Int$(\widetilde{y})$) features $O(HT)$ (resp. $O(T^2H)$) variables, where H is the number of groups of identical units.

For the ramp-constrained MUCP, inequalities (1.11)–(1.12) enforcing ramp-constraints are added to formulations $F(x,u)$, $W(x,u)$, $F(x,u,y)$ and $LF(x,u)$. Aggregated formulation A-$(\widetilde{x},\widetilde{u})$ can no longer be used, as its solutions cannot be disaggregated [50]. Note also that in this context, Start-up-ready inequalities are adjoined to $LF(x,u)$, but Shut-down-ready inequalities cannot.

In formulation Int$(\widetilde{y})$, the production limit constraint (5.13) is always included in inequalities (5.18) defining feasible productions. In the ramp-constrained case, the ramp constraints (5.14)–(5.17) are also included.

All experiments are performed using Cplex 12.8 C++ API on 8 threads of a PC with a 64 bit Intel Core i7-6700 processor running at 3.4GHz, and 32 GB of RAM memory. The UCP instances are solved until optimality (defined within 10^{-7} of relative optimality tolerance) or until the time limit of 3600 seconds is reached.

7.5.2 Instances

We generate 2-peak-demand MUCP instances as described in Section 1.2.4. The ramp-constrained MUCP instances considered are the same as in the non-ramp-constrained case, with additional ramp characteristics $RU^j = \frac{P^j_{max}-P^j_{min}}{3}$, $RD^j = \frac{P^j_{max}-P^j_{min}}{2}$ and $SU^j = SD^j = P^j_{min}$.

In order to determine which symmetry-breaking technique performs best with respect to the number of rows and columns of matrices in feasible set $\mathscr{X}$, we consider various instance sizes $n \in \{20,30,60\}$ and $T \in \{48,60\}$, and various symmetry factors $F \in \{2,3,4\}$. For each size (n,T) and symmetry factor $F \in \{2,3,4\}$, we generate a set of 20 instances. Symmetry factor $F = 4$ is not considered for instances with a small number n of units ($n = 20$ or 30), as it leads to very small sets of identical units.

Table 7.1 provides some statistics on the instances characteristics. For each instance, a group is a set of two or more units with the same characteristics. Each unit which has not been duplicated is a singleton. The first and second entries column-wise are the number of singletons and groups. The third entry is the average group size and the fourth entry is the maximum group size. Each entry row-wise corresponds to the average value obtained over 20 instances with same size (n,T) and same symmetry factor F.

7.5.3 Results for the MUCP

Tables 7.2, 7.3 and 7.4 provide, for each formulation and each group of 20 instances:

#opt:	Number of instances solved to optimality,
#nodes:	Average number of nodes,
gap:	Average optimality gap,
CPU time:	Average CPU time in seconds.

Note that a sign “-" in the column entry corresponding to the CPU time means that no instance could be solved within the time limit.

Size (n,T)	Sym. factor	Nb singl.	Nb groups	Av. group size	Group max. size
(20,48)	$F = 3$	1.25	4.90	3.96	5.75
	$F = 2$	0.75	3.20	6.45	8.75
(20,96)	$F = 3$	0.90	4.75	4.08	5.60
	$F = 2$	0.75	3.45	5.93	8.65
(30,48)	$F = 3$	1.10	5.35	5.51	9.45
	$F = 2$	0.25	3.85	8.30	12.60
(30,96)	$F = 3$	0.40	5.25	5.97	8.65
	$F = 2$	0.55	4.05	7.59	11.40
(60,48)	$F = 4$	0.80	7.70	7.86	13.20
	$F = 3$	0.55	5.80	10.90	17.80
	$F = 2$	0.20	4.75	13.90	23.80
(60,96)	$F = 4$	0.60	7.90	7.79	13.20
	$F = 3$	0.30	5.95	10.50	16.60
	$F = 2$	0.20	4.35	14.80	24.90

Table 7.1: Instance characteristics

It is clear that aggregated (x,u) formulation A-$(\widetilde{x},\widetilde{u})$ outperforms by far all the other formulations. This could be explained by the reduced size of aggregated formulation A-$(\widetilde{x},\widetilde{u})$, but also by the good performance of Cplex on ILP featuring integer variables (with bounds greater than 1). This efficiency will certainly be preserved any time the integer decomposition property holds for an (x,u) formulation of the UCP. Aggregated interval formulation Int$(\widetilde{y})$ is on average one or even two order of magnitude slower than $F(x,u)$, $F(x,u,z)$ and $LF(x,u)$. Formulations $F(x,u,z)$ and $W(x,u)$ are always outperformed by $F(x,u)$ and $LF(x,u)$. Formulations $F(x,u)$ and $LF(x,u)$ are quite comparable on $(n,T) = (20,48)$ instances. Interestingly, on $(n,T) = (20,96)$ instances, $LF(x,u)$ is better than $F(x,u)$. Otherwise, when n is larger ($i.e.$, $n \geq 30$), and when $T = 96$ or when $F = 2$, $F(x,u)$ outperforms $LF(x,u)$. On the opposite, when the horizon size is smaller ($i.e.$, T=48) and when $F \in \{3,4\}$, formulation $LF(x,u)$ outperforms $F(x,u)$ for $n = 30$ and $n = 60$. These two results may be due to Cplex's internal symmetry-detection and symmetry-breaking techniques, as in previous versions of Cplex (namely version 12.6.1), $LF(x,u)$ always outperformed $F(x,u)$.

7.5.4 Results for the ramp-constrained MUCP

Recall that aggregated formulation A-$(\widetilde{x},\widetilde{u})$ can no longer be used in this context.

Tables 7.5 and 7.6 provide, for each formulation and each group of 20 instances, the exact same column entries as those in Tables 7.2, 7.3 and 7.4.

First note that the ramp constraints make the MUCP instances much harder to solve by Cplex in general, as the CPU times in Tables 7.5 and 7.6 are much larger than those in Tables 7.2, 7.3 and 7.4. For example, the integrality gap is on average 10 times larger for ramp-constrained problems on $(n,T) = (60,48)$ and $F = 2$ instances.

Instances		Formulation	#opt	#nodes	gap (%)	CPU time
(20,48)	$F=2$	$F(x,u)$	20	1271	0	2.6
		A-$(\widetilde{x},\widetilde{u})$	20	0	0	0.2
		Int$(\widetilde{y})$	16	205667	0.00502	781.6
		$W(x,u)$	20	4809	0	13.7
		$F(x,u,z)$	20	3838	0	23.2
		$LF(x,u)$	20	1915	0	6.6
	$F=3$	$F(x,u)$	20	806	0	2.6
		A-$(\widetilde{x},\widetilde{u})$	20	0	0	0.3
		Int$(\widetilde{y})$	18	152948	0.00157	572.1
		$W(x,u)$	20	1600	0	4.4
		$F(x,u,z)$	20	683	0	6.7
		$LF(x,u)$	20	271	0	3.5
(20,96)	$F=2$	$F(x,u)$	20	148942	0	267.3
		A-$(\widetilde{x},\widetilde{u})$	20	0	0	0.7
		Int$(\widetilde{y})$	6	13180	9.36857	2977.5
		$W(x,u)$	18	110644	0.00015	459.6
		$F(x,u,z)$	18	118877	0.00018	497.8
		$LF(x,u)$	19	52881	0.00013	215.1
	$F=3$	$F(x,u)$	18	29418	0.03271	376.7
		A-$(\widetilde{x},\widetilde{u})$	20	2360	0	8.2
		Int$(\widetilde{y})$	7	32859	0.16855	2574.8
		$W(x,u)$	19	79864	0.00061	357.1
		$F(x,u,z)$	18	39694	0.00092	458.3
		$LF(x,u)$	19	19831	0.00085	229.1

Table 7.2: Performance indicators relative to the comparison of six formulations for MUCP instances with symmetries and $n=20$

Formulation Int$(\widetilde{y})$ is still the less efficient formulation. It does not solve to optimality any instance with $n>20$ but one. Moreover, on $n=30$ instances, and on $(n,T)=(60,96)$ instances, the root node cannot be processed at all within the time limit for formulation Int$(\widetilde{y})$; the number of nodes explored is 0 and the optimality gap is 100%.

Formulation $LF(x,u)$ is more efficient than all considered formulations. Formulation $LF(x,u)$ is able to solve a larger number of instances to optimality than all considered formulations, on all instances classes but $(n,T)=(20,96)$ $F=2$ instance class. In particular, $LF(x,u)$ manages to solve to optimality two of the large-size instances ($i.e.$, $(n,T)=(60,96)$), while other formulations do not reach optimality on any of these instances. Moreover, formulation $LF(x,u)$ solves 52 instances to optimality among the 80 instances with $n=30$, while $F(x,u,z)$ or $F(x,u)$ (resp. $W(x,u)$) only manages to solve to optimality 18 (resp. 24) of them. Among the 80 instances with $n=20$, formulation $LF(x,u)$ solves 57 instances to optimality, while $F(x,u,z)$ (resp. $W(x,u)$, $F(x,u)$) only manage to solve to optimality 49 (resp. 42, 34) of them. Formulation Int$(\widetilde{y})$ solves to optimality

Instances		Formulation	#opt	#nodes	gap (%)	CPU time
(30,48)	$F=2$	$F(x,u)$	20	3207	0	5.5
		A-$(\widetilde{x},\widetilde{u})$	20	0	0	0.2
		Int$(\widetilde{y})$	13	615979	0.00211	1483.4
		$W(x,u)$	18	67553	0.00018	370.7
		$F(x,u,z)$	18	41669	0.00021	391.5
		$LF(x,u)$	20	3894	0	19.4
	$F=3$	$F(x,u)$	19	157903	0.00001	189.1
		A-$(\widetilde{x},\widetilde{u})$	20	6	0	0.4
		Int$(\widetilde{y})$	13	267844	0.00772	1548.7
		$W(x,u)$	19	108689	0.00002	283.9
		$F(x,u,z)$	19	30450	0	249.5
		$LF(x,u)$	20	6044	0	18.5
(30,96)	$F=2$	$F(x,u)$	20	39633	0	131.2
		A-$(\widetilde{x},\widetilde{u})$	20	21	0	0.5
		Int$(\widetilde{y})$	0	21783	2.00212	-
		$W(x,u)$	18	54860	0.00029	679.8
		$F(x,u,z)$	18	20062	0.00083	758.5
		$LF(x,u)$	19	17040	0	299.5
	$F=3$	$F(x,u)$	20	15653	0	53.5
		A-$(\widetilde{x},\widetilde{u})$	20	215	0	0.6
		Int$(\widetilde{y})$	0	30267	4.38968	-
		$W(x,u)$	13	324913	0.00009	1400.7
		$F(x,u,z)$	17	123943	0.00003	907.1
		$LF(x,u)$	19	146639	0.00003	299.6

Table 7.3: Performance indicators relative to the comparison of six formulations for MUCP instances with symmetries and $n=30$

only 20 of them. Formulation $LF(x,u)$ also globally improves the solving time. For example, on instances of size $(n,T)=(60,48)$ and $F=3$, the average CPU time of formulation $LF(x,u)$ is 1450 seconds, while this number increases to 3422 (resp. 2527, 2689) for $F(x,u)$ (resp. $F(x,u,z)$, $W(x,u)$).

As there is an important variability in the computation time for instances with same size (n,T) and same F, we introduce the improvement score. For given formulations F_1 and F_2, the *improvement score* I of F_1 with respect to F_2 is as follows.

$$I = 2\frac{CPU(F_2)-CPU(F_1)}{CPU(F_2)+CPU(F_1)}$$

The improvement score I is a performance ratio comparing formulation CPU times pairwise.

Table 7.7 provides, for each formulation $F \in \{\, F(x,u,z),\, LF(x,u)\,\}$, the average improvement score of F with respect to $F(x,u)$ on each group of 20 instances. Formulation Int$(\widetilde{y})$ is not included as on most instance groups, it solves no instance to optimality.

Instances		Formulation	#opt	#nodes	gap (%)	CPU time
(60,48)	$F=2$	$F(x,u)$	19	60498	0	317.3
		A-$(\widetilde{x},\widetilde{u})$	20	69	0	0.2
		Int$(\widetilde{y})$	8	231718	0.00547	2345.3
		$W(x,u)$	17	43468	0.00008	825.5
		$F(x,u,z)$	19	6886	0	1234.4
		$LF(x,u)$	18	20969	0.00007	727.6
	$F=3$	$F(x,u)$	19	154845	0.00005	308.2
		A-$(\widetilde{x},\widetilde{u})$	20	17	0	0.4
		Int$(\widetilde{y})$	2	432040	0.01018	3287.8
		$W(x,u)$	17	82309	0.00007	602.5
		$F(x,u,z)$	19	20455	0.00005	674.7
		$LF(x,u)$	20	2362	0	87.3
	$F=4$	$F(x,u)$	17	326005	0.00022	587.4
		A-$(\widetilde{x},\widetilde{u})$	20	79	0	0.3
		Int$(\widetilde{y})$	9	298344	0.00672	2188.4
		$W(x,u)$	20	107657	0	349.9
		$F(x,u,z)$	18	52481	0.00008	893.3
		$LF(x,u)$	20	1222	0	32.0
(60,96)	$F=2$	$F(x,u)$	17	186561	0.00013	732.7
		A-$(\widetilde{x},\widetilde{u})$	20	0	0	0.3
		Int$(\widetilde{y})$	2	52352	0.08025	3421.5
		$W(x,u)$	8	202197	0.00064	2443.1
		$F(x,u,z)$	7	25805	0.00056	2875.7
		$LF(x,u)$	13	147301	0.00027	1850.6
	$F=3$	$F(x,u)$	12	831214	0.00044	1765.1
		A-$(\widetilde{x},\widetilde{u})$	20	98	0	0.6
		Int$(\widetilde{y})$	1	6201	17.59109	3586.3
		$W(x,u)$	6	287697	0.00092	2603.2
		$F(x,u,z)$	5	86252	0.00264	3051.5
		$LF(x,u)$	9	457416	0.00066	2190.5
	$F=4$	$F(x,u)$	16	297595	0.00020	906.0
		A-$(\widetilde{x},\widetilde{u})$	20	39	0	0.9
		Int$(\widetilde{y})$	1	6093	44.51052	3426.2
		$W(x,u)$	7	417944	0.00044	2566.1
		$F(x,u,z)$	9	59442	0.00084	2498.6
		$LF(x,u)$	10	386902	0.00026	1902.3

Table 7.4: Performance indicators relative to the comparison of six formulations for MUCP instances with symmetries and $n = 60$

Instances		Formulation	#opt	#nodes	gap (%)	CPU time
(20,48)	$F = 2$	$F(x,u)$	9	667974	0.00916	2061.6
		Int$(\widetilde{y})$	9	22583	0.04910	2426.1
		$W(x,u)$	10	232589	0.01115	1965.2
		$F(x,u,z)$	11	139493	0.00991	1840.4
		$LF(x,u)$	16	242096	0.00189	980.4
	$F = 3$	$F(x,u)$	13	634436	0.00296	1424.7
		Int$(\widetilde{y})$	3	7239	5.07207	3243.3
		$W(x,u)$	16	314447	0.00440	1295.9
		$F(x,u,z)$	18	102717	0.00226	998.0
		$LF(x,u)$	20	30014	0	132.8
(20,96)	$F = 2$	$F(x,u)$	5	702415	0.00776	2781.9
		Int$(\widetilde{y})$	3	10148	0.02754	3188.6
		$W(x,u)$	4	233582	0.02584	3058.1
		$F(x,u,z)$	8	61384	0.00681	2556.5
		$LF(x,u)$	6	160150	0.00718	2675.6
	$F = 3$	$F(x,u)$	7	989738	0.00644	2470.2
		Int$(\widetilde{y})$	5	16776	10.06768	3109.4
		$W(x,u)$	5	198137	0.01466	2725.6
		$F(x,u,z)$	12	87375	0.00424	1819.7
		$LF(x,u)$	15	186018	0.00565	1794.7
(30,48)	$F = 2$	$F(x,u)$	4	354029	0.01803	2924.7
		Int$(\widetilde{y})$	0	0	100	-
		$W(x,u)$	7	210032	0.01100	2535.4
		$F(x,u,z)$	4	71467	0.02547	2969.0
		$LF(x,u)$	15	219655	0.00204	1341.8
	$F = 3$	$F(x,u)$	6	379482	0.01213	2676.9
		Int$(\widetilde{y})$	0	0	100	-
		$W(x,u)$	10	240767	0.00698	1931.4
		$F(x,u,z)$	5	107609	0.01623	2736.1
		$LF(x,u)$	16	191113	0.00219	965.7
(30,96)	$F = 2$	$F(x,u)$	3	390666	0.00463	3069.8
		Int$(\widetilde{y})$	0	0	100	-
		$W(x,u)$	4	121205	0.00755	3130.1
		$F(x,u,z)$	5	46869	0.00918	3107.7
		$LF(x,u)$	9	315503	0.00238	2263.5
	$F = 3$	$F(x,u)$	5	460304	0.00324	2927.0
		Int$(\widetilde{y})$	0	0	100	-
		$W(x,u)$	3	211303	0.00465	3130.5
		$F(x,u,z)$	4	61994	0.00455	3059.7
		$LF(x,u)$	12	183633	0.00077	1852.9

Table 7.5: Performance indicators relative to the comparison of five formulations for ramp-constrained MUCP instances with symmetries and $n = 20, 30$

Instances		Formulation	#opt	#nodes	gap (%)	CPU time
(60,48)	$F=2$	$F(x,u)$	1	757017	0.00309	3437.6
		Int$(\tilde{y})$	0	7919	0.03078	-
		$W(x,u)$	4	203485	0.00285	3046.2
		$F(x,u,z)$	6	66272	0.03746	2839.8
		$LF(x,u)$	5	569546	0.00126	2710.6
	$F=3$	$F(x,u)$	1	850192	0.00268	3422.5
		Int$(\tilde{y})$	1	8300	5.18195	3523.9
		$W(x,u)$	6	192656	0.00245	2689.3
		$F(x,u,z)$	9	40680	0.00397	2527.5
		$LF(x,u)$	14	493254	0.00040	1450.2
	$F=4$	$F(x,u)$	7	870666	0.00243	2582.4
		Int$(\tilde{y})$	0	1236	25.95157	-
		$W(x,u)$	10	295149	0.00095	1971.9
		$F(x,u,z)$	14	33574	0.00053	1623.1
		$LF(x,u)$	15	459142	0.00027	1043.8
(60,96)	$F=2$	$F(x,u)$	0	120125	0.01262	-
		Int$(\tilde{y})$	0	0	100	-
		$W(x,u)$	0	23851	0.05190	-
		$F(x,u,z)$	0	3813	0.52855	-
		$LF(x,u)$	0	52226	0.01125	-
	$F=3$	$F(x,u)$	0	144265	0.01490	-
		Int$(\tilde{y})$	0	0	100	-
		$W(x,u)$	0	50841	0.01815	-
		$F(x,u,z)$	0	6404	0.03476	-
		$LF(x,u)$	0	83335	0.01311	-
	$F=4$	$F(x,u)$	0	230935	0.00956	-
		Int$(\tilde{y})$	0	0	100	-
		$W(x,u)$	0	92298	0.01063	-
		$F(x,u,z)$	0	9616	0.01589	-
		$LF(x,u)$	2	150692	0.00656	3467.7

Table 7.6: Performance indicators relative to the comparison of five formulations for ramp-constrained MUCP instances with symmetries and $n = 60$

Formulation $LF(x,u)$ outperforms all other formulations. In particular, even if on $(20,96)$ and $F=2$ instances the average CPU time of $LF(x,u)$ is slightly higher than $F(x,u,z)$, the average improvement score of $LF(x,u)$ is more important. This reveals that $LF(x,u)$ has a larger CPU time than $F(x,u,z)$ on instances for which the difference in CPU time is not very significant with respect to Cplex's CPU time. On the opposite, $LF(x,u)$ has small CPU time on instances on which this difference in CPU time represents an important improvement. Note that formulation $W(x,u)$ appears to perform better than $F(x,u,z)$ on $T=48$ instances. Recall that $W(x,u)$ is only partial symmetry-breaking. Thus, when T is smaller, the number of feasible columns featuring a given number of 1-entries is also smaller. On the opposite, when $T=96$, the number of one-entries is not a very discriminating indicator among symmetric columns. Therefore $W(x,u)$ is not able to break as much symmetries, and $F(x,u,z)$ globally performs better.

For example, on $(n,T)=(60,48)$, $F=4$ instances, the improvement score of $LF(x,u)$ is 109%, while it is 45.8% for $F(x,u,z)$ and 27.9% for $W(x,u)$. On $(n,T)=(30,48)$, $F=3$ instances, the improvement score of $LF(x,u)$ is 114%, while it is -19.3% for $F(x,u,z)$ and 25.3% for $W(x,u)$. On $(n,T)=(20,96)$, $F=3$ instances, this number increases to 47.6% for $LF(x,u)$ (resp. 22.9% for $F(x,u,z)$, -11.9% for $W(x,u)$).

Instances		Improvement score w.r.t. $F(x,u)$		
		$W(x,u)$	$F(x,u,z)$	$LF(x,u)$
(20,48)	$F=2$	-6.61%	-14.2%	83.7%
	$F=3$	-11.3%	-12.2%	111%
(20,96)	$F=2$	-13%	9.05%	23.9%
	$F=3$	-11.3%	22.9%	47.6%
(30,48)	$F=2$	20.8%	-17.8%	89.4%
	$F=3$	25.3%	-19.3%	114%
(30,96)	$F=2$	-15.8%	-19.4%	40.4%
	$F=3$	-11.8%	-7.63%	76.4%
(60,48)	$F=2$	26.5%	26.2%	47.8%
	$F=3$	35.5%	30.9%	104%
	$F=4$	27.9%	45.8%	109%
(60,96)	$F=2$	0%	0%	0%
	$F=3$	0%	0%	0%
	$F=4$	0%	0%	4.72%

Table 7.7: Improvement scores of formulations $W(x,u)$, $F(x,u,z)$ and $LF(x,u)$ w.r.t formulation $F(x,u)$ for ramp-constrained MUCP instances with symmetries

Conclusion

We propose a framework to build sub-symmetry-breaking inequalities, in order to handle the symmetries arising from a collection of sub-symmetric solution subsets. For each solution subset

Q considered, one additional variable z indicating "solution x belongs to Q" may be needed. Depending on the subset structure, variable z may only be a linear expression of variables x, and therefore does not need to be added to the model as an additional variable. The derived sub-symmetry-breaking inequalities are full symmetry-breaking under a mild condition. If this condition is not satisfied, a new collection of sub-symmetric subsets can be constructed such that the derived inequalities are full symmetry-breaking.

Our experimental results for the MUCP show that aggregation of the classical formulation is a very efficient technique to handle symmetries and sub-symmetries arising in the MUCP. When ramp constraints are taken into account in the MUCP, disaggregation is no longer possible. Our sub-symmetry-breaking inequalities can still be used and outperform all other formulations.

Sub-symmetry-breaking inequalities are always applicable as the solution subsets considered can capture the specific conditions under which the symmetries hold. On the opposite, aggregated formulations require specific conditions to be applicable.

One perspective is to use the proposed framework to derive new sub-symmetry-breaking inequalities for “ready to shut down" sub-symmetries in the ramp-constrained case. Another perspective is to apply the proposed framework to other problems featuring sub-symmetric solution subsets such as covering problems, or bin packing variants where one item can be placed in multiple bins. It would also be useful to study how the presented framework could be automated, so that sub-symmetric subsets are automatically detected and variables z automatically constructed.

Part III

Branch & Price for the MUCP

CHAPTER 8

DECOMPOSITION STRUCTURE

In this chapter, we study various Dantzig-Wolfe decomposition structures for the MUCP and the intra-site MUCP (IMUCP).

8.1 Motivations

There exist many variants of the UCP, depending on the constraint and cost structures considered. Although existing MILP solvers can efficiently handle large instances of difficult problems, some UCP variants still remain hard to solve for commercial solvers like Cplex 12.8. In particular, it has been shown in Chapter 4 that MUCP instances with $P^i_{min} = P^i_{max}$, $i \in \mathcal{N}$, are hard to solve even for small values of n ($n = 10$). The difficulty to solve these instances prefigures the difficulty to solve the UCP variant considered at EDF where the units have finite power outputs.

Moreover, many variants of the UCP feature non-linearities that impair the resolution process. Decomposition frameworks are particularly useful to account for non-linearities as the non-linear aspects of the problem can be moved to the subproblems. Then the Lagrangian function can be maximized via linear programming techniques, while the subproblems are solved with dedicated algorithms such as dynamic programming. This modularity can be further exploited, for example to devise efficient parallel implementations of the decomposition. The lower bound on the optimal value obtained from the dual bound is greater than or equal to the linear relaxation value, and can be used in a Branch & Bound framework [68] for an exact resolution. Otherwise primal and dual information obtained from the resolution of the Lagrangian dual can be exploited to design efficient heuristics [5, 18], such as primal-proximal heuristics, augmented Lagrangian based approaches or Price & Branch heuristics where columns are generated at the root node only. A question would be to compare the various heuristics and to measure the impact of these

relaxations on the optimal value.

The UCP features various structures which have been exploited to devise Lagrangian decomposition schemes. The literature on the subject [12, 52] is large and cannot be reviewed here. Classically, Lagrangian decomposition is performed for the UCP so that the demand and reserve constraints are dualized, and each unit (or subset of units) is treated as a subproblem which can be solved independently [5, 68].

The resolution of the UCP at EDF relies on such a decomposition, where each thermal unit is treated as a single subproblem. Recall that the intra-site constraints are satisfied if at most one unit per site Σ_k, $k \in \{1,...,K\}$, starts up at each time period t, *i.e.*, $\sum_{i\in\Sigma_k} u_t^i \leq 1$, for each $k \in \{1,...,K\}$ and $t \in \{1,...,T\}$. Coupling constraints such as intra-site constraints are currently not taken into account in the Lagragian decomposition performed at EDF. As demand and reserve constraints, intra-site constraints could be dualized, potentially hindering the resolution of the Lagrangian dual. To overcome this issue, and also to obtain a better Lagrangian bound, another possibility could be to adapt the decomposition structure to take such constraints into account, by treating each site as a subproblem. Given the results of Chapter 2, the resolution of such site subproblems, called P-IMUCP, will probably be more involved than the resolution of single unit subproblems.

Other decomposition approaches [47] consist in dualizing time coupling constraints such as min-up/down constraints so that the time horizon $\{1,...,T\}$ can be partitioned into several subsets of time periods treated as independent subproblems.

In Section 8.2, we study various decomposition structures such that the demand constraint is dualized and subsets of units are treated as subproblems. In Section 8.3, we describe another structure where the time coupling constraints are dualized, which amounts to treat each time period as a subproblem. Column generation algorithms are implemented to solve these various Dantzig-Wolfe master problems, and experimental results are compared in Section 8.4. Branch & Price (resp. Price & Branch) results are discussed in Section 8.5 (resp. 8.6). In Section 8.7, we give some perspectives about handling symmetries in the proposed decompositions. Note that all experimental details are given in Section 8.9 at the end of the chapter.

8.2 Unit subset decomposition of the IMUCP

In this section, the IMUCP is decomposed so that the demand constraint is dualized. The subproblems correspond to subsets of units coupled by intra-site constraints, or by other coupling inequalities.

When demand constraints and production limits are dualized in the IMUCP, the corresponding

master problem M_{DP} is the following:

$$\min_{\lambda^\pi, p_t^i} \quad \sum_{S\in\mathscr{S}} \left(\sum_{\pi\in\mathscr{P}^S} (c^\pi \lambda^\pi) + \sum_{t\in\mathscr{T}} \sum_{i\in S} c_p^i p_t^i + c_p^i P_{min}^i \Big(\sum_{\pi\in\mathscr{P}^s} a_t^{\pi,i} \lambda^\pi \Big) \right)$$

$$\text{s. t.} \quad \sum_{S\in\mathscr{S}} \sum_{i\in S} p_t^i + (P_{min}^i \sum_{\pi\in\mathscr{P}^S} a_t^{\pi,i} \lambda^\pi) \geq D_t \qquad \forall t \in \mathscr{T} \qquad (\mu_t)$$

$$p_t^i \leq (P_{max}^i - P_{min}^i)\Big(\sum_{\pi\in\mathscr{P}^s} a_t^{\pi,i} \lambda^\pi \Big) \qquad \forall i \in \mathscr{N},\ \forall t \in \mathscr{T} \qquad (v_t^i)$$

$$\sum_{\pi\in\mathscr{P}^S} \lambda^\pi = 1 \qquad \forall S \in \Sigma \qquad (\sigma^S)$$

$$\lambda_\pi \geq 0,\ p_t^i \geq 0 \qquad \forall i \in \mathscr{N},\ \forall t \in \mathscr{T}$$

where $a_t^{\pi,i}$ equals 1 if unit i is up at time t in up/down plan π and $\mathscr{P}^S$ is the set of up/down plans for site S. An up/down plan π for a site S indicates, for each unit i and each time step t, whether unit i is up or down at time t. The cost of plan π is denoted by c_π. Variable λ^π equals 1 if plan π is used. The constraint associated to dual variable μ_t is the demand constraint at time t. The constraint associated to v_t^i is the production limit, bounding the power generated by unit i at time t between P_{min}^i and P_{max}^i. The constraint associated to σ_S ensures that a single plan π is chosen for each site S.

For each site S, the corresponding column generation subproblem is to find a minimum reduced-cost plan. It can be written as the following ILP:

$$\min_{x,u} \quad -\sigma^S + \sum_{i\in S} \sum_{t=1}^{T} \Big(c_f^i + (c_p^i - \mu_t) P_{min}^i - (P_{max}^i - P_{min}^i) v_t^i \Big) x_t^i + c_0 u_t^i$$

$$\text{s. t.} \quad \sum_{t'=t-L^i+1}^{t} u_{t'}^i \leq x_t^i \qquad \forall i \in S,\ \forall t \in \{L+1,...,T\} \qquad (8.1)$$

$$\sum_{t'=t-\ell^i+1}^{t} u_{t'}^i \leq 1 - x_{t-\ell}^i \qquad \forall i \in S,\ \forall t \in \{\ell+1,...,T\} \qquad (8.2)$$

$$u_t^i \geq x_t^i - x_{t-1}^i \qquad \forall i \in S,\ \forall t \in \mathscr{T} \qquad (8.3)$$

$$\sum_{i\in S} u_t^i \leq 1 \qquad \forall t \in \{2,...,T\}$$

$$x_t^i, u_t^i \in \{0,1\} \qquad \forall i \in S,\ \forall t \in \mathscr{T}$$

Recall that min-up, min-down and up/start-up relationship inequalities (8.1), (8.2) and (8.3) are inequalities (1.2) – (1.4) introduced in [79].

8.2.1 Dualization of production constraints

We refer to constraints involving production variables as *production constraints*. Production limits or ramp constraints are examples of production constraints. If all production constraints are dualized, then the column generation subproblem only features variables x and u. Thus the plans generated in the subproblems are only up/down plans, and the production decisions are

taken in the Dantzig-Wolfe master problem. On the opposite, if some production constraints are not dualized, production decisions are also taken in the subproblems, potentially dramatically increasing the number of feasible solutions to the subproblem.

Therefore the question is whether dualizing production constraints alongside with demand constraint has an impact on the dual bound.

• **Non-ramp-constrained IMUCP** In the non-ramp-constrained case, it can easily be seen that it does not change the dual bound. Recall that M_{DP} is the Dantzig-Wolfe master problem where both the demand and production limits are dualized. Let M_D be the column generation master problem where only the demand is dualized.

Lemma 8.1. *The optimal value of M_D is equal to the optimal value of M_{DP}.*

Proof. Consider a solution $(\overline{x}, \overline{u}, \overline{p})$ to M_{DP}. Then, by definition, $(\overline{x}, \overline{u}) = \sum_{k=1}^{s} \lambda_k (x(k), u(k))$ where $\sum_{k=1}^{s} \lambda_k = 1$ and for each $k \in \{1, ..., s\}$, $(x(k), u(k))$ is a binary solution satisfying min-up/down and intra-site constraints. For each k, let

$$p(k)_t^i = \begin{cases} \frac{\overline{p}_t^i}{\overline{x}_t^i} & \text{if } x(k)_t^i = 1 \\ 0 & \text{otherwise} \end{cases}$$

Then $(x(k), u(k), p(k))$ is a solution with binary $x(k)$ and $u(k)$ satisfying min-up/down constraints, production limits and intra-site constraints. Moreover, $(\overline{x}, \overline{u}, \overline{p}) = \sum_{k=1}^{s} \lambda_k (x(k), u(k), p(k))$. Therefore from any solution $(\overline{x}, \overline{u}, \overline{p})$ to M_{DP} a same-cost solution to M_D can be constructed. The reverse is trivially true. ■

• **Ramp-constrained IMUCP** In the ramp-constrained case, the result does not hold anymore. Let M_{DPR} be the Dantzig-Wolfe master problem where demand, production limits and ramp-constraints are dualized. Then the optimal value of M_{DPR} is less than or equal to the optimal value of M_D. In the following example, the optimal value of M_{DPR} is strictly less than that of M_D.

Example 8.1. *Consider the following ramp-constrained MUCP instance with a single unit ($n = 1$) and $T = 3$, where $\ell^1 = L^1 = 1$, $P^1_{min} = 10$, $P^1_{max} = 30$, $RU^1 = 5$ and $SU^1 = 20$. Then solution $x = [0.5, 1, 1]$, $\rho = [10, 15, 20]$ satisfies ramp-up constraint* (1.11)*:*

$$\rho_t^1 - \rho_{t-1}^1 \leq RU^i x_{t-1}^1 + (SU^i - P^1_{min}) u_t^1$$

Solution (x, ρ) is therefore a solution to M_{DPR}. However, (x, ρ) is not feasible for M_D. Indeed, x is a convex combination of two integer solutions x_1 and x_2:

$$x = \frac{1}{2} x_1 + \frac{1}{2} x_2, \qquad \text{where } x_1 = [0, 1, 1] \text{ and } x_2 = [1, 1, 1]$$

If the production limit is not dualized, then any solution (x_1, ρ_1) *(resp.* (x_2, ρ_2)*) generated by the subproblem is such that*

$$\rho_1 \leq [0, 10, 15] \quad \textit{and} \quad \rho_2 \leq [20, 20, 20]$$

As $\rho > \frac{1}{2}\rho_1 + \frac{1}{2}\rho_2$*, solution* (x, ρ) *is thus not feasible for* M_D*.*

This shows that in the ramp-constrained case, leaving production constraints in the subproblems may improve the dual bound, at the expense of increasing the subproblems combinatorics. The problems considered in the following do not feature ramp-constraints, therefore production constraints will always be dualized.

8.2.2 Granularity of the unit-subset decomposition

Depending on which coupling constraints are dualized, the unit subsets corresponding to subproblems may contain a single unit or several units.

Unit decomposition If intra-site constraints are dualized alongside with demand constraints, the subproblem decomposes into n subproblems, *i.e.*, one subproblem per single unit. Then only min-up/min-down constraints remain in each subproblem. Therefore, the unit decomposition is a demand-coupling formulation (see Section 3.1), and by Theorem 3.1, the dual bound is less than or equal to the linear relaxation value of formulation $F^n(x, u)$. By Corollary 1.1, the dual bound is greater than or equal to the linear relaxation value. It follows that the dual bound is equal to the linear relaxation value.

In this particular case, the use of Lagrangian decomposition does not lie in the quality of the bound it provides but in its modularity. Non-linear start-up costs can for example be handled by dynamic programming through such a unit decomposition.

Site decomposition If intra-site constraints are not dualized, then the subproblem decomposes into K subproblems, *i.e.*, one subproblem per site. Then the site decomposition is not a demand-coupling formulation, as each subproblem features min-up/down and intra-site constraints. No complete linear description of this polytope is known, therefore, by Theorem 1.4, the dual bound is potentially a better bound than the linear relaxation value.

Residual demand decomposition Additional inequalities can be considered, in order to improve the dual bound, as well as to provide more information to the subproblems. For a given partition of the unit set into subsets S_1, ..., S_r, the units of each subset S_k must cover at least the residual demand $D_t - \sum_{j \notin S_k} P^j_{max}$ (as defined in Section 3.3). The corresponding inequality is the *residual demand* constraint:

$$\sum_{i \in S_k} P^i_{max} x^i_t \geq D_t - \sum_{j \notin S_k} P^j_{max}, \qquad \forall k \in \{1, ..., s\}$$

This inequality is redundant in $(F^n_{x,u})$ formulation of the (I)MUCP. However, in a decomposition framework where only demand constraints and production limits are dualized, residual demand constraints remain in the subproblems. If for any site Σ, there exists k such that $\Sigma \subseteq S_k$, the subproblem decomposes into s subproblems, one per unit subset S_k. Each subproblem features min-up/down, residual demand and possibly intra-site constraints. If subsets S_k, $k \in \{1,...,s\}$ are chosen so that $D_t - \sum_{j \notin S_k} P^j_{max} > 0$, then the residual demand is a knapsack constraint. Therefore the residual demand decomposition is not a demand-coupling formulation and the dual bound is potentially a better bound than the linear relaxation value.

8.2.3 Start-up decomposition

Reducing the combinatorics of the subproblems is commonly known to have a positive impact on the convergence of the column generation algorithm. One option is to restrict the subproblem's solution set to a particular solution subset, and at the same time use *exchange vectors*, as defined in [90]. Exchange vectors are additional variables in the master problem. The resulting formulation remains valid, as all solutions can be obtained using the combination of exchange vectors with columns coming from restricted subproblems.

Note that in the LP dual of the master problem, the exchange vectors can be seen as additional cuts coupling dual variables and reducing the feasible dual domain. This corresponds to the stabilization technique pointed out in [87].

In the case of the unit-subset decomposition of the IMUCP, the idea is that only start-up decisions are taken in the subproblems, *i.e.*, the subproblems solutions are restricted to start-up plans. In order to obtain up/down plans we artificially decide to complete each start-up plan so that each unit is down during exactly ℓ^i time periods before each start-up. More formally, the up/down plans generated will be such that when a unit i starts up at time t_1, it remains up until time $t_2 - \ell^i$, where t_2 is the next start-up of unit i after t_1.

Exchange vectors are defined as variables z^i_t indicating that unit i is down at time t, while the chosen plan π indicated that unit i was up at time t. In other words, if unit i can be shut down before the artificial shut-down decision in plan π, the new shut-down decision will be taken in the master problem using variables z^i_t.

The corresponding Dantzig-Wolfe master problem is as follows:

$$\min_{\lambda^\pi} \quad \sum_{S \in \Sigma} \left(\sum_{\pi \in \mathscr{P}^S} (c^\pi \lambda^\pi) + \sum_{t \in \mathscr{T}} \sum_{i \in S} (c^i_p p^i_t + c^i_p P^i_{min}(-z^i_t + \sum_{\pi \in \mathscr{P}^s} a^{\pi,i}_t \lambda^\pi) - c^i_f z^i_t) \right)$$

$$\text{s. t.} \quad \sum_{S\in\Sigma}\sum_{i\in S}\left(p_t^i + P_{min}^i(-z_t^i + \sum_{\pi\in\mathscr{P}^S} a_t^{\pi,i}\lambda^\pi)\right) \geq D_t \qquad \forall t\in\mathscr{T} \qquad (\mu_t)$$

$$p_t^i \leq (P_{max}^i - P_{min}^i)(-z_t^i + \sum_{\pi\in\mathscr{P}^S} a_t^{\pi,i}\lambda^\pi) \qquad \forall i\in\mathscr{N},\ \forall t\in\mathscr{T} \qquad (\nu_t^i)$$

$$\sum_{\pi\in\mathscr{P}^S}\lambda^\pi = 1 \qquad \forall S\in\Sigma \qquad (\sigma^S)$$

$$z_t^i + \sum_{t'=t-L^i+1}^{t}\sum_{\pi\in\mathscr{P}^S} b_{t'}^{\pi,i}\lambda^\pi \leq \sum_{\pi\in\mathscr{P}^S} a_t^{\pi,i}\lambda^\pi \qquad \forall S\in\Sigma,\ \forall i\in S,\ \forall t\in\{L^i+1,...,T\} \qquad (\zeta_t^i)$$

$$\sum_{\pi\in\mathscr{P}^S}(a_t^{\pi,i} - a_{t-1}^{\pi,i} - b_t^{\pi,i})\lambda^\pi + z_{t-1}^i - z_t^i \leq 0 \qquad \forall S\in\Sigma,\ \forall i\in S,\ \forall t\in\{L^i+1,...,T\} \qquad (\theta_t^i)$$

$$z_t^i \leq \sum_{\pi\in\mathscr{P}^S} a_t^{\pi,i}\lambda^\pi \qquad \forall S\in\Sigma,\ \forall i\in S,\ \forall t\in\mathscr{T} \qquad (\xi_t^i)$$

$$\lambda_\pi \geq 0,\ z_t^i \geq 0,\ p_t^i \geq 0 \qquad \forall S\in\Sigma,\ \forall i\in S,\ \forall t\in\mathscr{T}$$

where $b_t^{\pi,i}$ equals 1 if unit i starts up at time t in up/down plan π. The constraint associated to dual variable ζ_t^i corresponds to the min-up constraint and the constraint associated to θ_t^i corresponds to constraint (1.4) linking start-up to up/down decisions. The constraint associated to ξ_t^i ensures that z_t^i is greater than zero only if unit i is up at time t in the chosen plan π.

For each site S, the subproblem is to find a minimum reduced-cost start-up plan. A start-up plan corresponds to an up/down plan where the down time between two up periods of unit i is always equal to ℓ^i. The corresponding subproblem is as follows.

$$\min_{x,u} -\sigma^S + \sum_{i\in S}\sum_{t=1}^{T}\left((c_f^i + (c_p^i - \mu_t)P_{min}^i - (P_{max}^i - P_{min}^i)\nu_t^i - \xi_t^i - \theta_t^i + \theta_{t+1}^i + \zeta_t^i)x_t^i + (c_0 + \theta_t^i + \sum_{t'=t}^{t+L^i-1}\zeta_{t'}^i)u_t^i\right)$$

$$\text{s. t.} \quad \sum_{t'=t-L^i+1}^{t} u_{t'}^i \leq x_t^i \qquad \forall i\in S,\ \forall t\in\{L+1,...,T\}$$

$$\sum_{t'=t-\ell^i+1}^{t} u_{t'}^i = 1 - x_{t-\ell}^i \qquad \forall i\in S,\ \forall t\in\{\ell+1,...,T\}$$

$$u_t^i \geq x_t^i - x_{t-1}^i \qquad \forall i\in S,\ \forall t\in\mathscr{T}$$

$$\sum_{i\in S} u_t^i \leq 1 \qquad \forall t\in\{2,...,T\}$$

$$x_t^i, u_t^i \in \{0,1\} \qquad \forall i\in S,\ \forall t\in\mathscr{T}$$

8.2.4 Resolution of the subproblems

In the unit decomposition case, each subproblem is a 1-unit MUCP, and thus can be polynomially solved (see Section 1.2.6), by linear or dynamic programming techniques.

In the site (resp. residual demand) decomposition case, each subproblem is a P-IMUCP (resp. IMUCP), which has been shown NP-hard in the strong sense in Chapter 2, Section 2.5 (resp. Section 2.1). However, for a fixed number n of units, a polynomial dynamic programming scheme is proposed in Chapter 2, Section 2.3.

8.3 Time decomposition of the IMUCP

In this section, the IMUCP is decomposed so that the time coupling constraints (*i.e.*, min-up/min-down constraints) are dualized. Each subproblem corresponds to one given time period t where all units are coupled via the demand constraint at time t.

The corresponding master problem is the following:

$$
\begin{aligned}
\min_{\lambda^\pi, u_t^i, p_t^i} \quad & \sum_{t\in\mathcal{T}} \left(\sum_{\pi\in\mathscr{P}^t} c^\pi \lambda^\pi + \sum_{i\in\mathcal{N}} c_0^i u_t^i \right) & & \\
\text{s. t.} \quad & u_t^i \geq \sum_{\pi\in\mathscr{P}^t} a^{\pi,i}\lambda^\pi - \sum_{\pi\in\mathscr{P}^{t-1}} a^{\pi,i}\lambda^\pi & \forall i\in\mathcal{N},\ \forall t\in\{2,\dots,T\} & (\mu_t^i) \\
& \sum_{t'=t-L^i+1}^{t} u_{t'}^i \leq \sum_{\pi\in\mathscr{P}^t} a^{\pi,i}\lambda^\pi & \forall i\in\mathcal{N},\ \forall t\in\{L^i+1,\dots,T\} & (\nu_t^i) \\
& \sum_{t'=t-\ell^i+1}^{t} u_{t'}^i \leq 1 - \sum_{\pi\in\mathscr{P}^{t-\ell^i}} a^{\pi,i}\lambda^\pi & \forall i\in\mathcal{N},\ \forall t\in\{\ell^i+1,\dots,T\} & (\xi_t^i) \\
& \sum_{i\in S} u_t^i \leq 1 & \forall S\in\Sigma,\ \forall t\in\{2,\dots,T\} & (\eta_t^S) \\
& \sum_{\pi\in\mathscr{P}^t} \lambda^\pi = 1 & \forall t\in\mathcal{T} & (\sigma^t) \\
& \lambda_\pi \geq 0, u_t^i \geq 0 & \forall i\in\mathcal{N},\ \forall t\in\mathcal{T}, \forall \pi\in\mathscr{P}^t &
\end{aligned}
$$

where $a^{\pi,i}$ is equal to 1 if unit i is up in plan π, and $\mathscr{P}^t$ is the set of up/down plans for the n units at time t. Note that $\mathscr{P}^t$ differs from $\mathscr{P}^S$ defined in Section 8.2. Variable λ^π equals 1 if plan π is used. The constraint associated to dual variable μ_t^i corresponds to constraint (1.4) linking start-up to up/down decisions. The constraint associated to ν_t^i (resp. ξ_t^i) is the min-up (resp. min-down) constraint.

For each $t\in\mathcal{T}$, the subproblem is to find an up/down plan at time t satisfying D_t. The corresponding ILP is the following:

$$
\begin{aligned}
\min_{x} \quad & -\sigma_t + \sum_{i\in\mathcal{N}} c_p^i p_t^i + (c_f^i + c_p^i P_{min}^i + \mu_t^i - \mu_{t+1}^i + \xi_{t+\ell^i}^i - \nu_t^i) x_t^i & & \\
\text{s. t.} \quad & P_{min}^i x_t^i \leq p_t^i \leq P_{max}^i x_t^i & \forall i\in\mathcal{N} & \\
& \sum_{i\in\mathcal{N}} p_t^i \geq D_t & & \\
& x_t^i \in\{0,1\},\ p_t^i\in\mathbb{R} & \forall i\in\mathcal{N} &
\end{aligned}
$$

where if $t+\ell^i > T$, then $\xi_{t+\ell^i}^i = 0$ and if $t\leq L^i$, then $\nu_t^i = 0$.

Note that the time decomposition is not a demand-coupling formulation, therefore the dual bound is potentially a better bound than the linear relaxation value.

Drawback In the presence of heterogeneous units featuring intricate technical constraints, the master problem of the time decomposition may be hard to manage. If some of these difficult

constraints are relaxed, the dual bound obtained from the time decomposition is still a valid lower bound on the optimal solution.

8.3.1 Dualization of production constraints

Note that all production constraints are dualized in the time decomposition of the IMUCP, thus production variables appear exclusively in the subproblems. Therefore, only the up/down decisions taken in the subproblem are transmitted to the master problem. The associated production decisions can be directly deduced from the up/down decisions.

In the case of the ramp-constrained IMUCP, the ramp-constraints must be dualized, as they induce a dynamic coupling between two time steps. Then, there are two possible time decomposition structures. One option is to generate only up/down plans from the subproblems. Then the demand constraint must appear in both the subproblem and in the master problem. Note that in this case, the demand constraint impacts the discrete decisions (x,u) in the subproblem, while it impacts the continuous decisions p in the master problem. Another possible structure is to generate up/down and production plans in the subproblem. In this case, the demand constraint appears only in the subproblem.

8.3.2 Time decomposition with interval up-set inequalities

In order to improve the dual bound provided by the time decomposition, interval up-set inequalities, written as a function of λ variables, can be added to the master problem. Let $C \subset \mathcal{N}$ be a subset of units, with $i \in C$, and let $\mathcal{I} = \{t_0, ..., t_1\} \subset \mathcal{T}$ be a time interval of length less than or equal to L^i, *i.e.*, $t_1 - t_0 \leq L^i$. The interval up-set inequality can be written with λ variables as follows.

$$\alpha_{\mathcal{I}}(C) + \sum_{t=t_0+1}^{t_1} u_t^i \leq \sum_{\pi \in \mathscr{P}^{t_1}} a^{\pi,i} \lambda^\pi + \sum_{j \in C \setminus \{i\}} \left(\sum_{\pi \in \mathscr{P}^{t_0}} a^{\pi,j} \lambda^\pi + \sum_{t'=t_0+1}^{t_1} u_t^j \right) \tag{8.4}$$

Since the demand constraint is not dualized, note that for each time $t \in \{1, ..., T\}$ solutions to the master problem satisfy all inequalities arising from the knapsack polytope at time t. In particular static up-set inequalities (see Chapter 3, Section 3.4.1) are automatically satisfied.

8.3.3 Resolution of the subproblem

We consider the classical case with integer demands $D_t \in \mathbb{N}$, $t \in \{1, ..., T\}$.

When $P^i_{min} = P^i_{max}$, $i \in \mathcal{N}$, the subproblem can be identified to a 0-1 knapsack problem with constraint

$$\sum_{i=1}^{n} P^i_{max}(1 - x^i) \leq -D_t + \sum_{i=1}^{n} P^i_{max}$$

where the decision $(1 - x^i)$ amounts to not committing unit i. The classical $O(n(-D_t + \sum_{i=1}^{n} P^i_{max}))$ dynamic programming algorithm for the knapsack problem can therefore be used. Preliminary

numerical experiments show that on average, this algorithm is 1000 times faster than Cplex 12.8 used as with default settings.

When $P^i_{min} \neq P^i_{max}$, $i \in \mathcal{N}$, the subproblem can be solved by the pseudo-polynomial dynamic programming scheme described in Section 2.2. For the sake of brevity, we will only consider instances with $P^i_{min} = P^i_{max}$, $i \in \mathcal{N}$ in the next sections.

8.4 Experimental results relative to dual bounds

In this section, we compare, in terms of dual bounds and convergence, the resolution by column generation of the Dantzig-Wolfe master problems presented in Sections 8.2 and 8.3.

Due to the various structures to compare, we explicit the content of this section to ease the reading. In Section 8.4.1, we give experimental details alongside with a summary table relative to the results of Unit, Site, ResD and Time decomposition. In Section 8.4.3, we compare decompositions Unit, Site and ResD in order to determine the most appropriate granularity for the unit subset decomposition. On this basis, we assess the impact of the start-up decomposition in Section 8.4.4. In Section 8.4.5, we analyze the results of the column generation algorithm for the time decomposition. In Section 8.4.5, we study the impact of interval up-set inequalities in the time decomposition.

Note that all details are given in Tables 8.7 to 8.14 in Section 8.9 at the end of the chapter.

8.4.1 Experimental settings

Column generation algorithms are implemented within the SCIP 5.0.1 [33] framework, on 8 threads of a PC with a 64 bit Intel Xeon(R) E3-1240 processor running at 3.5GHz, and 32 GB of RAM memory. The instances are solved until optimality (defined within 10^{-7} of relative optimality tolerance) or until the time limit of 3600 seconds is reached.

Decomposition structures compared We will present experimental results comparing the column generation algorithm for the following decomposition structures:

Unit	unit decomposition,
Unit-SU	start-up decomposition, where each subproblem corresponds to a single unit,
Site	site decomposition,
Site-SU	start-up decomposition, where each subproblem corresponds to a site,
ResD	residual demand decomposition,
Time,	time decomposition.
Time+I	time decomposition with separation of interval up-set inequalities.

The site decomposition is performed only for IMUCP instances, as it relies on the presence of intra-site constraints.

Instances In Chapter 4, TPR-100 instances featuring a 2-peak per day demand have proved to be very hard to solve. Moreover, preliminary experiments indicate that intra-site constraints have only little impact on the optimal value on instances featuring 2-peak per day demands. The impact of intra-site constraints seems to be more important on instances with random demand values. Therefore, for both the MUCP and the IMUCP, column generation algorithms are run on six sets of ten TPR-100 instances, featuring

- 2-peak per day demand, (n,T) = (20,24), (20,48)
- Random valued demand, (n,T) = (20,24), (20,48)

Note that size $(n,T) = (20,48)$ instances are difficult instances, as Cplex 12.8 is not able to solve them to optimality within a time limit of one hour.

MUCP and IMUCP instances with same size and same demand type are identical. The only difference is that intra-site constraints are enforced only for IMUCP instances. The unit subsets considered in residual demand decomposition correspond to sites, for both MUCP and IMUCP, even though for the MUCP intra-site constraints are not taken into account.

Note that we did not consider symmetrical instances in these experiments. Handling symmetries is left for future work (see Section 8.7 for some insights).

Resolution of the subproblems For practical reasons we choose to use:

- Cplex 12.8 with default settings to solve the subproblems of Unit, Site and ResD decompositions,
- the classical dynamic programming scheme for the 0-1 knapsack problem to solve the time decomposition subproblem (as described in Section 8.3.3).

Unit and Site subproblems could be solved by dynamic programming. As here we are mainly interested in the quality of the dual bound, and the number of iterations to reach it, the implementation of an efficient dynamic programming algorithm for these problems is out of the scope of this work.

Comparison of Unit, Site, ResD and Time We compare column generation algorithms for Unit, Site, ResD and Time decompositions.

Table 8.1 presents, for each set of 10 instances with same size and demand type:

Opt. val. the average optimal value,

and for each decomposition structure,

#iter	the average number of iterations,
CPU	the average CPU time (in seconds),
Dual b.	the average dual bound,
Dual b. Δ LR	the average difference between the dual bound and the linear relaxation value
Dual b. Δ Cplex b.	the average difference between the dual bound and the bound obtained with Cplex's cuts.

Note that the linear relaxation value of the (x,u) formulation is not given, as it is exactly the dual bound obtained with the unit decomposition.

The average dual bounds are of the order of 10^5 while the average differences "Dual b. Δ LR" or "Dual b. Δ Cplex b." are from the order of 10 to the order of 10^4. In order to preserve numerical precision, we consider absolute values in the tables instead of ratios.

The average optimal values for $(20,48)$ 2-peak-demand instances are not given, because only 2 out of 10 (for both the MUCP and the IMUCP) can be obtained by Cplex within a time limit of one hour.

Discussions on Table 8.1 are dispatched in Subsection 8.4.2, 8.2.2 and 8.4.5. Each subsection focuses on a particular point.

Note that instance-wise results are given in Tables 8.7 to 8.14, in Section 8.9 at the end of the chapter.

8.4.2 Impact of intra-site constraints

Recall that MUCP and IMUCP instances are the same. The only difference is that intra-site constraints are enforced in the IMUCP case. Therefore, by comparing same size and same demand profile MUCP and IMUCP instances, we can assess the impact of intra-site constraints over the linear relaxation and the optimal values.

2-peak-demand instances Comparing the results for MUCP and IMUCP instances in Table 8.1, the intra-site constraints have only a marginal impact on the optimal value, as well as on the linear relaxation value. Referring to Tables 8.7 to 8.10, on 8 instances over 20, the bound obtained by Cplex's cut even decreases when intra-site constraints are added. It proves that when intra-site constraints come into play, Cplex does not recognize the problem's structure as efficiently.

Random-demand instances The intra-site constraints have more impact in the random-demand context. Indeed, as shown in Table 8.1, the average difference between optimal values of IMUCP and MUCP instances is in the order of 5000 on $(20,24)$ instances, while this difference is in the order of 100 in the 2-peak case.

		2-peak-demand				Random-demand			
		MUCP		IMUCP		MUCP		IMUCP	
		(20, 24)	(20,48)	(20, 24)	(20,48)	(20, 24)	(20,48)	(20, 24)	(20,48)
Opt. val.		297540	×	297802	×	418297	816537	423792	828320
#iter	Unit	393	1517	465	1936	192	489	343	1190
	Site	-	-	846	2960	-	-	275	746
	ResD	866	3258	812	2917	255	663	227	578
	Time	12345	30674	12035	31534	92062	183835	80157	196622
CPU	Unit	1	4	1	4	1	2	1	3
	Site	-	-	3	15	-	-	2	7
	ResD	4	17	3	17	2	9	2	9
	Time	2	11	2	12	36	117	26	143
Dual b.	Unit	288828	615519	288948	615718	409435	800343	415024	813037
	Site	-	-	288948	615719	-	-	415035	813085
	ResD	288828	615519	288948	615719	411850	804748	417147	816793
	Time	295645	627619	295705	627711	417367	814395	422572	825684
Dual b.	Unit	0	0	0	0	0	0	0	0
Δ	Site	-	-	0	1	-	-	10	48
LR	ResD	0	0	0	1	2415	4406	2123	3756
	Time	6817	12100	6757	11993	7931	14053	7548	12647
Dual b.	Unit	-6468	-11325	-6377	-11240	-7733	-14124	-7792	-12674
Δ	Site	-	-	-6377	-11240	-	-	-7782	-12626
Cplex b.	ResD	-6468	-11325	-6377	-11240	-5318	-9718	-5670	-8918
	Time	348	775	380	752	199	-71	-244	-27

* Note that the CPU time of Time decomposition is not to be compared with that obtained with other decompositions (see paragraph "Resolution of the subproblems" in Section 8.4.1)

"-" indicates that the corresponding decomposition structure was not used on the corresponding instance set

"×" indicates that instances could not be solved to integer optimality by Cplex within time limit

Table 8.1: Summary table relative to column generation results of Tables 8.7 to 8.14

Indeed, when the demand is random, the variation of the demand from time t to $t+1$ can be much more important than in the 2-peak case. Therefore, if the demand increases from t to $t+1$, it is likely that several units must start up at time $t+1$. If these units are located on the same site, then intra-site constraints will prevent the simultaneous start-ups, thus modifying the optimal value.

8.4.3 Granularity of the unit subset decomposition

In this section, we compare the dual bounds obtained and the number of column generation iterations in order to define the appropriate granularity for the unit subset decomposition, depending on the demand profile.

2-peak-demand instances As shown in Table 8.1, the unit decomposition has in general the lowest CPU time (less than one second on $(n,T) = (20,24)$ instances, and around 3 seconds on $(n,T) = (20,48)$ instances). Site and ResD decompositions have higher CPU times (around 3 seconds on $(n,T) = (20,24)$ instances, and around 15 seconds on $(n,T) = (20,48)$ instances). Unit, Site and ResD decompositions spend most of their time in the pricing problem solved by Cplex. The difference between Unit and the other unit-subset decompositions (namely Site and ResD) is that the subproblems of the latter decompositions (respectively the P-IMUCP and the IMUCP) are more difficult to solve by Cplex than Unit's subproblem.

As shown in Table 8.1, Site and ResD decompositions have similar number of column generation iterations, while the unit decomposition is the quickest (in terms of iteration number) to converge, by a factor 2 on average. Note that the number of iterations increases by a factor 2 to 4 when T increases from 24 to 48.

Referring to Tables 8.7 to 8.10, the number of priced variables follows a similar pattern.

As shown in Table 8.1, the dual bound obtained by the site decomposition is almost equal to the linear relaxation value. The dual bound obtained by ResD decomposition is not better than the one obtained by site decomposition.

Random-demand instances Interestingly, while the Unit decomposition has the smallest column generation iteration number in the 2-peak case, when the demand is random the picture is different. On MUCP instances, Table 8.1 shows that ResD decomposition requires slightly more iterations than Unit to converge. On the opposite, on IMUCP instances, Site and ResD decompositions converge significantly faster than Unit (by a factor 2 on average).

On $(n,T) = (20,24)$, Site and ResD require a similar number of iterations to reach convergence, but on larger instances ($i.e.$, $(n,T) = (20,48)$), ResD converges faster (in terms of iterations) than Site.

While in the 2-peak-demand case the ResD decomposition did not provide better bounds than Unit or Site formulations, when the demand is random, the dual bounds obtained by ResD decomposition are better for both the MUCP and the IMUCP.

For example, as shown in Table 8.1, the ResD bound improves the linear relaxation value by an additive term of 4000 on average, on $(n,T) = (20,48)$ instances random demand.

These bounds are however not as good as the ones obtained with Cplex's cuts or with the time decomposition.

Conclusion It appears that for 2-peak-demand (I)MUCP instances, the appropriate granularity for a unit subset decomposition structure is the unit decomposition. Indeed, the column generation algorithm converges faster in this case, while the dual bound obtained is not worst than in other unit subset decompositions. For random-demand IMUCP instances, the site decomposition is better than the Unit decomposition, and the ResD decomposition is the best, in terms of dual bound and convergence. For random-demand MUCP instances, the unit decomposition converges

slightly faster, but the dual bound obtained by the ResD decomposition is better. Therefore, it appears that ResD is also the most appropriate decomposition structure for the random-demand MUCP.

8.4.4 Start-up decomposition

Various unit subset decompositions have been compared in Section 8.4.3. The question is whether the start-up decomposition could improve the convergence of the column generation algorithms for these decomposition structures.

We implement the start-up decomposition structure with the most appropriate granularity according to the instances' demand profile. For 2-peak-demand instances, we compare Unit-SU, the start-up decomposition (implemented within a unit decomposition structure) to the unit decomposition. For random-demand instances, the site decomposition converges faster, and provides a better dual bound on the optimal value, than the unit decomposition. Therefore we compare Site-SU, the start-up decomposition (implemented within a site decomposition structure) to the site decomposition on these instances. Note that only start-up decisions are taken in the subproblems, and the residual demand constraints relies on the up/down decisions. Thus we did not consider implementing the start-up decomposition with ResD granularity.

As the presence of intra-site constraints did not impact much the column generation algorithms, we only considered IMUCP instances for both 2-peak and random demand instances.

Table 8.2 presents the corresponding results, for each set of 10 IMUCP instances with same demand type, and for each decomposition structure. The column entries are the following

#iter	the average number of iterations,
CPU	the average CPU time (in seconds),
Dual b.	the average dual bound,

Instance-wise results are presented in Tables 8.15 and 8.16, in Section 8.9 at the end of the chapter.

		2-peak-demand	
		(20,24)	(20,48)
#iter	Unit	465	1936
	Unit-SU	2708	8045
CPU	Unit	1	4
	Unit-SU	1	5
Dual b.	Unit	288948	615718
	Unit-SU	288948	615718

(a)

		Random-demand	
		(20,24)	(20,48)
#iter	Time	275	746
	Time-SU	1883	5047
CPU	Time	2	7
	Time-SU	2	10
Dual b.	Time	415035	813085
	Time-SU	415024	813045

(b)

Table 8.2: Summary table relative to column generation results for Start-up decompositions presented in Tables 8.15 and 8.16.

The hope was that by reducing the combinatorics of the subproblem, the start-up decomposition would help to converge faster. Even though, as shown in Tables 8.15 and 8.16, the number of variables is slightly reduced in the Unit-SU (resp. Site-SU) decomposition compared to the Unit (resp. Site) decomposition, it appears that the number of iterations increases by a factor of 4 on average on $(n,T) = (20,24)$ and on $(n,T) = (20,48)$ instances (see Table 8.2). Therefore the start-up decompositions do not appear to be of interest within these unit subset decomposition structures.

8.4.5 Time decomposition

The results of the column generation algorithm for the time decomposition can be found in Tables 8.7 to 8.14. Table 8.1 summaries the results.

First note that as shown in Tables 8.7 to 8.14, for the time decomposition, most of the CPU time is spent in solving the master problem. Indeed, the pricing problem is solved very efficiently by dynamic programming (recall from Paragraph "Resolution of the subproblems" that this cannot be compared to unit-subset decomposition cases).

2-peak-demand instances The time decomposition has a lower CPU time than Site and ResD decomposition, with a CPU time in the order of 2 seconds on $(n,T) = (20,24)$ instances, and in the order of 10 seconds on $(n,T) = (20,48)$ instances (see Table 8.1).

As shown in Table 8.1, the number of column generation iterations performed in the time decomposition is always higher (by a factor from 10 to 100) than in other decompositions. As for unit subset decompositions, the number of iterations increases by a factor of 2 at least when T increases from 24 to 48. Referring to Tables 8.7 to 8.10, the number of priced variables follows a similar pattern.

The best dual bound is obtained with the time decomposition. This bound is always far better than the one obtained with the other decompositions.

Indeed, as shown in Table 8.1, the time decomposition bound improves the linear relaxation value by an additive term greater than 10000 (resp. 7000) on each MUCP and IMUCP $(n,T) = (20,48)$ (resp. $(20,24)$) instance.

Interestingly, the bounds obtained are even better than the lower bounds computed via Cplex's own cuts, on each instance (except instance 10 of size $(20,48)$, see Tables 8.7 to 8.10). As the demand constraint is not dualized, the bound obtained from this decomposition is the same as the bound obtained when all facets of the knapsack polytopes at time t, $t \in \mathcal{T}$, are added to the linear relaxation. We pointed out in Chapter 4 that many useful static up-set (*i.e.*, extended cover) inequalities were not automatically added by Cplex 12.8. The present results confirm this observation.

Random-demand instances Random demand values tend to give much more importance to dynamic constraints, *i.e.*, min-up/down constraints. Therefore, as shown in Table 8.1, for a given size (n,T), the number of column generation iterations of the time decomposition is multiplied by a factor of 10 on average when random demands are considered instead of 2-peak demands. Correspondingly, the CPU time of the time decomposition is also higher than in the 2-peak case, as it is in the order of 30 (resp. 100) seconds for $(n,T) = (20,24)$ (resp. (20,48)) instances.

The dual bounds obtained by the time decomposition are always better than the bounds obtained by other decomposition structures. As in the 2-peak case, the time decomposition bound improves the linear relaxation value by an additive term greater than 10000 on each MUCP and IMUCP $(n,T) = (20,48)$ instance.

The time decomposition bounds are often better than Cplex's bound, but it is not always the case. Referring to Tables 8.11 to 8.14, the time decomposition provides a better bound than Cplex's cuts on 12 MUCP instances (resp. on 8 IMUCP instances) over the 20 considered. Dynamic constraints having now more importance, the relative impact of static up-set inequalities appears to be reduced.

Conclusion On random-demand instances, time decomposition provides a good bound, sometimes better than the one obtained with Cplex's cuts. On 2-peak-demand instances, the time decomposition provides a much better dual bound than Cplex's cuts or any other decomposition structure does. In both cases, the bound's quality comes at the expense of an increase in the number of iterations.

8.4.6 Time decomposition with interval up-set inequalities

The time decomposition is the decomposition structure providing the best lower bound on the optimal value. The dual bound obtained is even better than the linear relaxation value obtained with the cuts generated by Cplex. In this section, static interval up-set inequalities are separated in the master problem in order to further improve this dual bound.

We perform column generation iterations until no column is found. Then we apply separation algorithms until no violated interval up-set inequalities can be found. The process is iterated until no improving columns nor inequality is found. The separation algorithm used is the one described in Chapter 4. Recall that since interval up-set inequalities can be expressed in original (x,u) variables, adding these inequalities only modifies the cost structure in the subproblem.

For MUCP and IMUCP instances, we compare Time and Time+I, the time decomposition with separation of interval up-set inequalities. For the record, very few interval up-set inequalities are found on $(n,T) = (20,24)$ instances. Therefore we focus here on $(n,T) = (20,48)$ instances.

Table 8.3 presents, for each set of 10 instances with same demand type,

Δ(Time+I,Time)	the average difference between the dual bounds of Time+I and Time
Δ(Time+I,Cplex)	the average difference between the dual bounds of Time+I and the bound obtained with Cplex's cuts

and for each decomposition structure,

#iter	the average number of iterations,
Dual b.	the average dual bound,

		2-peak-demand		Random-demand	
		MUCP	IMUCP	MUCP	IMUCP
#iter	Time	30674	31534	183835	196622
	Time+I	30765	31604	184188	197120
Dual b.	Time	627619	627711	814395	825684
	Time+I	627628	627719	814430	825795
Δ(Time+I,Time)		8.96	8.59	35.31	111.08
Δ(Time+I,Cplex)		783.84	760.98	-36.18	83.92

Table 8.3: Summary table relative to column generation results for time decomposition with interval up-set inequalities presented in Tables 8.17 and 8.18.

Instance-wise results can be found in Tables 8.17 and 8.18.

2-peak-demand instances For MUCP and IMUCP instances with 2-peak demands, few interval up-set inequalities are found (from 0 to 6 per instance, see Table 8.17). This is enough to increase the dual bound by an additive term of order 8 on average, as shown in Table 8.3. Note also that the number of column generation iterations does not increase significantly when interval up-set inequalities are separated. As the inequalities are separated once the column generation algorithm has converged, the number of iterations cannot decrease from Time to Time+I.

Random-demand instances As was the case with 2-peak-demand instances, the convergence of the column generation is not impacted by the separation of interval up-set inequalities.

For MUCP instances, only few interval up-set inequalities are found (5 in total for the ten instances, see Table 8.18). Interestingly, many more inequalities are found for IMUCP instances (25 in total). Violated interval up-set inequalities are found in only 3 instances out of 10, for both MUCP and IMUCP instance sets. However, the resulting dual bound is much improved by these inequalities, as it is increased by an additive term of order 30 to 100. On $(20,48)$ IMUCP instances, while the average dual bound provided by Time was not as good as the bound provided by Cplex's cuts (see Table 8.1), when interval up-set are added the average bound obtained with Time+I is better than Cplex's, as shown in Table 8.3.

Conclusion On instances with larger horizon size (*i.e.*, $T = 48$), interval up-set inequalities enable to improve even further the dual bound obtained with the time decomposition, at no additional cost as the number of iterations remains quite similar.

8.5 Experimental results relative to Branch & Price & Cut

As for 2-peak-demand instances, the most appropriate granularity for the unit subset decomposition is the unit decomposition, we compare Branch & Price (& Cut) algorithms on 2-peak-demand instances for Unit, Time and Time+I decomposition structures.

For random-demand instances, the most appropriate granularity for the unit subset decomposition is the residual demand decomposition, thus on these instances we compare Time and Time+I decompositions to ResD.

Table 8.4 presents the results of default Cplex on each set of 10 instances with same size and demand profile. It provides:

#solved	the number (out of 10) of instances solved to optimality,
Nodes	the average number of nodes,
CPU	the average CPU time (in seconds),

		2-peak-demand				Random-demand			
		MUCP		IMUCP		MUCP		IMUCP	
		(20, 24)	(20,48)	(20, 24)	(20,48)	(20, 24)	(20,48)	(20, 24)	(20,48)
Cplex	#solved	10	2	10	2	10	10	10	10
	Nodes	10210	159400	7500	161600	1217	2427	1094	2903
	CPU	93	3232	83	3444	6	18	7	29

Table 8.4: Summary of Default Cplex results on the instances considered

As the scope of this chapter is to compare various decomposition structures, the goal is not yet to implement Branch & Price algorithms that could be competitive with commercial solvers like Cplex. Our implementation could be hugely improved for example by finely-tuned branching rules, stabilization techniques, or fast dynamic programming algorithms for the subproblems. This is why in the following we will not compare our Branch & Price statistics to Cplex.

Branch & Price algorithms are implemented in the same experimental settings as described in Section 8.4. At the root node, the column generation algorithms are initialized with columns obtained using Cplex's primal heuristics (corresponding to the best solution Cplex obtains at the root node of the B&B).

The branching is performed on up/down decisions, *i.e*, the branching disjunction has the form

$$\sum_{\pi \in \mathscr{P}} a_t^{\pi,i} \lambda^\pi = 1 \quad \vee \quad \sum_{\pi \in \mathscr{P}} a_t^{\pi,i} \lambda^\pi = 0$$

At each node, the most fractional $\sum_{\pi \in \mathscr{P}} a_t^{\pi,i} \lambda^\pi$ is chosen for branching.

A time limit of 3600 seconds is set.

8.5.1 First results on small-size instances

First, preliminary experiments are run on small-size, *i.e.*, $(n,T) = (10,24)$, MUCP and IMUCP instances featuring 2-peak or random demand profiles.

Tables 8.19 to 8.22 provide Branch & Price results on these small-size instances. They present, for each instance and each decomposition structure:

id	the instance number,
#nodes	the number of nodes,
IUP	the number of interval up-set inequalities separated
#col	the number of columns generated,
CPU	the CPU time (in seconds) of the Branch & Price,
Gap	the optimality gap
Primal b.	the best integer solution found within time limit.

The main result is that time decompositions (Time and Time+I) outperform by far unit subset decompositions, *i.e.*, unit decomposition for 2-peak-demand instances and ResD decomposition for random demand instances. Indeed, the unit subset decompositions manage to solve to optimality only 2 instances out of the 40 small-size instances, even though the number of nodes explored is 100 to 10000 times larger than that of the time decompositions.

Therefore in the following, only Time and Time+I are performed on instances of size $(20,24)$ and $(20,48)$.

8.5.2 Results on larger instances

Table 8.5 presents the Branch & Price results for Time and Time+I for each set of 10 instances. The column entries are the same as in Table 8.4, with additional entries:

#IUP	the average number of interval up-set found
I_N	the average node improvement score of Time+I w.r.t. Time
I_{CPU}	the average CPU time improvement score of Time+I w.r.t. Time

The improvement scores are defined in Section 4.2. For each instance set, the node (resp. CPU) improvement score in computed for the subset of instances where both (resp. at least one of the) decompositions reach optimality.

Instance-wise results can be found in Tables 8.23 to 8.26.

None of the size (20,48) 2-peak-demand instances can be solved to optimality (within time limit) using time decompositions, as shown in Table 8.5. Cplex only manages to solve to optimality 2 of them, as shown in Table 8.4.

		2-peak-demand				Random-demand			
		MUCP		IMUCP		MUCP		IMUCP	
		(20, 24)	(20,48)	(20, 24)	(20,48)	(20, 24)	(20,48)	(20, 24)	(20,48)
#solved	Time	10	0	9	0	8	5	9	5
	Time+I	10	0	8	0	8	5	9	4
Nodes	Time	7402	5831	11294	4825	528	622	386	453
	Time+I	6538	5204	9837	4518	531	525	408	413
CPU	Time	827	3600	1431	3600	890	2805	818	2645
	Time+I	855	3600	1463	3600	882	2398	890	2513
#IUP		33	81	47	77	1	5	1	7
I_N		16.82 %	-	3.16 %	-	0.50 %	16.76 %	5.2 %	9.04 %
I_{CPU}		4.90 %	-	-3.98 %	-	2.26 %	20.60 %	-1.4 %	8.36 %

"-" indicates that none of the instances has been solved to optimality

Table 8.5: Summary of the Branch & Price results presented in Tables 8.23 to 8.26

Interestingly, Table 8.5 shows that interval up-set inequalities improve the number of nodes, on all sets of instances where optimality can be reached. While this improvement in the number of nodes does not show on the CPU time on the small ($i.e.$, (n,T) = (20,24)) instances, when T increases, the cost of solving a large LP at each node is well compensated by the better bounds obtained with interval up-set inequalities.

For random-demand instances, fewer interval up-set are found, but the improvement they induce is still significant, for example the CPU time is improved by 20.6% on average on (20,48) random-demand MUCP instances.

8.6 Experimental results relative to Price & Branch heuristic

In this section, we take advantage of the Branch & Price framework to derive a Price & Branch heuristic. The Price & Branch algorithm is such that columns are generated at the root node only, and Branch & Bound is applied on the resulting formulation.

Price & Branch algorithms are implemented in the same experimental settings as described in Section 8.5. Once the columns are generated within SCIP framework, the resulting LP is given to Cplex to solve.

Preliminary results indicate that Price & Branch algorithms based on unit-subset decompositions are not competitive, at least without further improvement. Indeed, the branching process converges very slowly, and intermediate solutions are not as good as the one obtained with Cplex in a shorter time. Thus we compare

CplexHeur Cplex with a short time limit,

TimeP&B, Price & Branch in the time decomposition framework

The idea is to assess the quality of the solution obtained within a very short time, therefore

a global time limit is set to 180 seconds (resp. 30 seconds) for both methods on (20,48) (resp. (20,96)) instances.

Table 8.6 presents the result for 2-peak-demand IMUCP instances of size $(n,T) = (20,96)$ as well as (20,48). It provides

id	the instance number,
#col	the number of columns generated,
Dual b.	the best lower bound found within the time limit.
Primal b.	the best integer solution found within the time limit.
CPU-Price	the CPU time of the column generation algorithm at the root
CPU	the total CPU time (in seconds)

Note that we do not present the results for random instances, as on such instances the column generation algorithm converges too slowly to enable Price & Branch to run fast enough to be competitive with CplexHeur.

As shown in Table 8.6 for the Time Price & Branch heuristic, the most CPU consuming step is the column generation at the root node. In terms of solutions quality, Price & Branch sometimes finds much better solutions than Cplex, especially on large instances. For example, on instances 2, 8 and 10 (resp. 3 and 5), Price & Branch finds a solution which costs at least 2000 (resp. 1000) less than Cplex's solution.

8.7 Perspectives on symmetry-breaking in Dantzig-Wolfe reformulations

In practice, identical units are often located on the same site, inducing symmetries in the master problem or in the subproblems, depending on the decomposition considered. Symmetries arising in the master problem impair the Branch & Price process, as they do for Branch & Bound algorithms in general. Breaking them would therefore be helpful towards the integer resolution of the Dantzig-Wolfe reformulation.

Handling symmetries arising in the subproblems would accelerate the resolution of the subproblem, and could play a role towards the convergence of the column generation algorithm, by reducing the number of solutions generated by the subproblem.

8.7.1 Handling symmetries in unit-subset decompositions

To handle symmetries arising in the master problem of Dantzig-Wolfe reformulations, it is shown in [89] that master variables corresponding to identical subproblems can be aggregated, in the case of pure integer programs. As our problem features continuous variables, this aggregation result does not apply.

	id	#col	Dual b.	Primal b.	CPU-Price	CPU
$(n,T) = (20,48)$						
CplexHeur	1	-	670021	682979	-	30.12
TimeP&B	1	1395	670940.8	682257	22.11	22.72
CplexHeur	2	-	666681.6	682456.2	-	30.13
TimeP&B	2	1857	668217.4	681454	28.92	30.78
CplexHeur	3	-	517789.7	524282.8	-	30.15
TimeP&B	3	1978	516953.5	526660	26.46	27.91
CplexHeur	4	-	664033	680069.9	-	30.09
TimeP&B	4	1750	665131.6	680941	26.65	28.98
CplexHeur	5	-	698243.9	704227.3	-	30.12
TimeP&B	5	1459	698809.5	706069	22.9	23.3
CplexHeur	6	-	553488.1	565152.1	-	30.15
TimeP&B	6	1900	553409.6	564081	26.75	27.96
CplexHeur	7	-	641659.3	653270.5	-	30.09
TimeP&B	7	1790	642197.1	652197	29.03	29.73
CplexHeur	8	-	642417.5	650893.6	-	30.14
TimeP&B	8	1549	642386.8	650479	24.31	25.54
CplexHeur	9	-	657908	662109	-	30.12
TimeP&B	9	1587	658045.9	661964	24.27	24.79
CplexHeur	10	-	560957.7	569067.5	-	30.21
TimeP&B	10	1717	560099.7	570260	22.93	24.71
$(n,T) = (20,96)$						
CplexHeur	1	-	975936	988753.4	-	181.2
TimeP&B	1	4097	975212.1	989776	93.93	104.2
CplexHeur	2	-	1511353	1540946	-	181.1
TimeP&B	2	4578	1512152	1538980	147.6	166.8
CplexHeur	3	-	1367876	1393661	-	181.1
TimeP&B	3	3580	1369440	1392510	94.28	101.6
CplexHeur	4	-	1215102	1238913	-	181.7
TimeP&B	4	3152	1215950	1239410	79.85	110.2
CplexHeur	5	-	1195142	1218029	-	180.8
TimeP&B	5	4377	1196509	1216800	140.4	174.6
CplexHeur	6	-	1399552	1432186	-	180.9
TimeP&B	6	3930	1400784	1431370	112	120.6
CplexHeur	7	-	1374570	1404107	-	181
TimeP&B	7	3480	1375282	1404220	91.31	96.55
CplexHeur	8	-	1283919	1312527	-	180.9
TimeP&B	8	4101	1285344	1310270	151.5	168.9
CplexHeur	9	-	1189979	1205344	-	181.1
TimeP&B	9	4074	1189686	1206330	99.11	118.6
CplexHeur	10	-	1175454	1198234	-	180.7
TimeP&B	10	3147	1176128	1196250	91.93	95.65

Table 8.6: Price and Branch – IMUCP instances – 2-peak demand

For the unit decomposition of the (I)MUCP, it is still possible to aggregate master variables λ and p corresponding to identical units. Aggregated solutions $(\overline{\lambda}, \overline{p})$ with integer $\overline{\lambda}$ can be disaggregated into solutions (λ, p) with integer λ as shown in [50].

For the unit decomposition of the ramp-constrained (I)MUCP, if production constraints are not dualized, then only variables λ appear in reformulation (DW). In this case, each variable λ corresponds to a feasible up/down and production plan $\pi = (x, p) \in \{0,1\}^{(n,T)} \times \mathbb{R}^{(n,T)}$ for a given unit. An interesting question is whether master variables λ can be aggregated in this context.

In any case, the aggregation of master variables λ prevents from branching on non-aggregated decisions (otherwise symmetries would be reintroduced).

When aggregation is not possible, or when flexibility with respect to the branching decisions must be preserved, sub-symmetry-breaking inequalities or orbitopal fixing for the full sub-orbitope can be used to handle symmetries arising in the master problem of the Dantzig-Wolfe reformulation.

When the decomposition is made along unit subsets (containing more than just one unit), symmetries arise in the subproblems featuring identical units. The question is how these symmetries can be exploited to solve the subproblems more efficiently, while avoiding generating symmetrical plans from the subproblems.

In the non-ramp-constrained case, aggregation of subproblems variables is possible. As previously, it will prevent from branching on non-aggregated decisions.

When aggregation of the subproblems' variables is not possible, as in the ramp-constrained case, an interesting perspective is to handle symmetries within the subproblem's dedicated resolution technique.

8.7.2 Handling symmetries in time decomposition

In the presence of identical units, the Dantzig-Wolfe reformulation obtained by time decomposition also features symmetric solutions.

For example, consider a solution λ. Then for any symmetry σ, permuting identical units, solution λ_σ is obtained from λ as follows:

$$\lambda^\pi_\sigma = \lambda^{\sigma^{-1}(\pi)}, \qquad \forall \pi \in \mathscr{P}^t, \forall t \in \mathscr{T}$$

i.e., for each time t, for $\pi \in \mathscr{P}^t$ such that $\lambda^\pi = 1$, the plan selected at time t in solution λ_σ is the permutation of plan $\sigma(\pi)$. Therefore λ^σ is feasible and has same cost than λ.

Sub-symmetry-breaking inequalities (7.6) and (7.7) can be used to handle such symmetries, replacing variables x^i_t by $\sum_{\pi \in \mathscr{P}^t} a^{\pi,i} \lambda^\pi$. Orbitopal fixing for the full orbitope can also be used, where instead of fixing variables directly to 0 or 1, sums of variables $\sum_{\pi \in \mathscr{P}^t} a^{\pi,i} \lambda^\pi$ would be equal

to 0 or 1. Such equalities can be used as cuts to the master problem. Instead, these equalities can also be propagated in order to fix some λ variables to 0 or 1.

8.8 Conclusion

We compare various decomposition structures for the (I)MUCP. The column generation algorithm converges quite fast on unit-subset decompositions, but the dual bound obtained is of poor quality. The time-based decomposition needs much more iterations to converge, but the dual bound it provides is better than that of the bound obtained with Cplex's cuts on many instances. Without further improvement of the lower bound, implemented Branch & Price algorithms based on unit-subset decompositions are completely outperformed by time decomposition Branch & Price. The latter features promising CPU times, which are significantly enhanced by interval up-set inequalities. The Price & Branch algorithm based on the time decomposition is a quick (less than 3 minutes) heuristic, providing good quality solutions compared to Cplex used with a 3-minute time limit.

A first perspective would be to study how to include ramp constraints in the studied decomposition structures. In particular, when production constraints are not dualized, the combinatorics of the subproblem highly increases in the ramp-constrained case, which may lead to convergence issues. If the production constraints are dualized, then the dual bound may be lower. However this may be compensated for by adding valid inequalities to the Dantzig-Wolfe master problem.

Another related perspective would be to define the appropriate granularity for time-based decompositions, so that each subproblem would correspond to an appropriate subset of time steps. It would also be useful to improve unit-subset-based decompositions by the addition of valid inequalities. The goal would be to catch up with the bounds obtained with time-based decompositions. For the latter decompositions, the column generation algorithm could be enhanced as well, in particular with stabilization techniques in order to reduce the number of iterations. More dedicated branching rules would improve the corresponding Branch & Price algorithm.

Another crucial question for the time decomposition framework is how to account for heterogeneous units featuring various technical constraints. In particular, the question is whether the time decomposition master problem can still be solved efficiently by LP solvers. If not, then the follow-up question is to what extent some technical constraints can be relaxed without impacting the quality of the dual bound provided.

Finally, even though no experimental results have been carried out for symmetrical instances, the symmetry-breaking techniques proposed in this thesis can be used in all studied decomposition structures.

8.9 Experimental tables

Column generation for Unit, Site, ResD and Time decompositions Column generation algorithms are compared for Unit, Site, ResD and Time decomposition. Results for 2-peak-demand instances are presented in Table 8.7 (resp. Table 8.9) for $(n,T) = (20,24)$ (resp. $(n,T) = (20,48)$) instances of the MUCP, and in Table 8.8 (resp. Table 8.10) for $(n,T) = (20,24)$ (resp. $(n,T) = (20,48)$) instances of the IMUCP.

Results for random-demand instances are presented in Table 8.11 (resp. Table 8.13) for $(n,T) = (20,24)$ (resp. $(n,T) = (20,48)$) instances of the MUCP, and in Table 8.12 (resp. Table 8.14) for $(n,T) = (20,24)$ (resp. $(n,T) = (20,48)$) instances of the IMUCP.

Tables 8.7 to 8.14 present, for each instance:

id	instance number,
#iter	number of column generation iterations,
#col	number of columns generated,
CPU	CPU time (in seconds) of the column generation,
M-CPU	CPU time (in seconds) spent in solving the master problem,
Dual b.	optimal value of the column generation master problem
CplexCuts	linear relaxation value of the IMUCP formulation without decomposition, once Cplex has added its own cuts,
Opt,	optimal value of the IMUCP, if computed in less than 3600 seconds by Cplex

Note that the linear relaxation value of the (x,u) formulation is not given, as it is exactly the dual bound obtained with the unit decomposition.

	id	#iter	#col	CPU	M-CPU	Dual b.	CplexCuts	Opt
Unit	1	306	247	0.77	0.01	321686.6		
ResD	1	760	229	3.57	0.09	321686.6	329026.5	332471.5
Time	1	18684	1194	4.18	4	329201.5		
Unit	2	454	309	1.11	0.01	308326.5		
ResD	2	1046	256	4.26	0.04	308326.5	314346.3	316585.4
Time	2	11057	715	1.53	1.36	315075.1		
Unit	3	355	268	0.79	0.01	231689.2		
ResD	3	765	212	2.55	0.03	231689.2	239899.1	242467.1
Time	3	15937	911	2.69	2.59	240191.5		
Unit	4	382	285	0.88	0	301511		
ResD	4	800	248	4.16	0.08	301511	308751.9	311586.1
Time	4	11089	892	1.93	1.82	309164.7		
Unit	5	404	315	0.93	0.02	247945.9		
ResD	5	976	271	6.44	0.06	247945.9	252727.3	254636
Time	5	6808	598	1.02	0.94	253171.3		
Unit	6	381	291	0.86	0.03	302242.6		
ResD	6	781	243	3.45	0.06	302242.6	308680.3	309777.4
Time	6	19261	975	3.48	3.22	308911.1		
Unit	7	387	258	0.85	0.03	273102.7		
ResD	7	878	272	3.81	0.04	273102.7	277760.9	280374.3
Time	7	8307	671	1.29	1.2	278176.1		
Unit	8	481	292	0.95	0.03	311345.7		
ResD	8	1042	275	4.34	0.13	311345.7	317900.1	319807.6
Time	8	8952	729	1.44	1.29	318158.6		
Unit	9	382	265	0.8	0.05	254416.8		
ResD	9	901	248	3.24	0.09	254416.8	261867.5	263334.5
Time	9	12419	830	2.16	2.01	262004.8		
Unit	10	397	293	1.01	0.04	336012.4		
ResD	10	708	233	2.4	0.05	336012.4	342002.5	344357.9
Time	10	10938	754	1.78	1.63	342391.2		

Table 8.7: Column generation – MUCP instances – (n,T) = (20,24), 2-peak per day demand

	id	#iter	#col	CPU	M-CPU	Dual b.	CplexCuts	Opt
Unit	1	341	256	0.86	0.08	321704.4		
Site	1	830	247	3.7	0.08	321704.4	329020.2	332471.5
ResD	1	829	247	5.79	0.05	321704.4		
Time	1	20879	1237	5.09	4.84	329201.5		
Unit	2	459	295	0.91	0.04	308326.5		
Site	2	1029	243	4.22	0.08	308326.5	314377.2	316859.1
ResD	2	850	238	3.71	0.09	308326.5		
Time	2	10962	748	1.6	1.47	315076.5		
Unit	3	440	257	0.66	0.02	231853.3		
Site	3	764	201	2.06	0.06	231853.3	239929.8	242612.1
ResD	3	654	197	2.24	0.03	231853.3		
Time	3	12531	831	2	1.85	240191.5		
Unit	4	443	269	0.94	0.05	301511		
Site	4	747	234	2.58	0.09	301511	308712.2	312081.8
ResD	4	740	225	3.56	0.06	301511		
Time	4	10263	870	2.01	1.9	309167.1		
Unit	5	557	324	0.94	0.06	247949.3		
Site	5	1037	253	2.53	0.07	247949.3	252704.2	254675.6
ResD	5	1059	258	3.34	0.06	247949.3		
Time	5	8260	645	1.21	1.14	253171.9		
Unit	6	480	286	0.93	0.06	302446		
Site	6	640	217	2.44	0.06	302446	308982.2	310647.6
ResD	6	669	219	2.95	0.06	302446		
Time	6	14815	970	2.74	2.56	309331.2		
Unit	7	417	255	0.72	0.04	273110.7		
Site	7	787	242	2.42	0.06	273110.7	277855.9	280374.3
ResD	7	735	229	2.98	0.07	273110.7		
Time	7	9111	703	1.46	1.36	278176.9		
Unit	8	514	298	0.98	0.05	311345.7		
Site	8	991	267	3.2	0.1	311345.7	317802.4	320128.2
ResD	8	998	251	4.45	0.03	311345.7		
Time	8	8814	728	1.46	1.3	318158.6		
Unit	9	469	273	0.78	0.03	255163.7		
Site	9	887	255	3.26	0.06	255163.7	262028.6	263334.5
ResD	9	850	262	3.72	0.04	255163.7		
Time	9	13577	766	2.13	1.97	262162.9		
Unit	10	528	287	0.85	0.03	336067		
Site	10	748	233	1.97	0.07	336067	341838.3	344836.8
ResD	10	736	215	2.08	0.04	336067		
Time	10	11139	753	1.92	1.8	342412.2		

Table 8.8: Column generation – IMUCP instances – (n,T) = (20,24), 2-peak per day demand

	id	#iter	#col	CPU	M-CPU	Dual b.	CplexCuts	Opt
Unit	1	2067	719	4.99	0.21	656370.1		
ResD	1	3586	662	16.04	0.44	656370.1	669861.6	-
Time	1	24375	1569	7.8	7.61	670940.8		
Unit	2	1770	670	4.25	0.14	656409.3		
ResD	2	2738	577	14.54	0.27	656409.3	666597.9	-
Time	2	30053	1801	12.54	12.22	668217.4		
Unit	3	942	452	2.35	0.07	503744.9		
ResD	3	2294	505	16.67	0.31	503744.9	516337.5	522116.9
Time	3	34952	2120	14.43	13.98	516953.5		
Unit	4	1424	619	3.63	0.16	652443.2		
ResD	4	3508	627	17.02	0.47	652443.2	664117.2	-
Time	4	33468	1834	11.64	11.33	665131.6		
Unit	5	1706	610	3.48	0.21	687042.8		
ResD	5	4678	677	33.08	0.68	687042.8	697968.4	-
Time	5	29300	1649	8.99	8.79	698809.5		
Unit	6	1280	517	3.01	0.12	542302.5		
ResD	6	1868	495	10.51	0.17	542302.5	552529.6	-
Time	6	38587	2188	15.18	14.79	553409.6		
Unit	7	1455	590	3.67	0.15	632669.9		
ResD	7	2857	563	10.6	0.3	632669.9	641500.1	-
Time	7	33713	1863	12.88	12.63	642197.1		
Unit	8	1499	562	2.82	0.13	629820.6		
ResD	8	3169	563	13.83	0.39	629820.6	641662.7	-
Time	8	26037	1735	8.94	8.66	642386.8		
Unit	9	1582	608	3.65	0.18	646706.9		
ResD	9	4479	666	26.17	0.67	646706.9	657462.2	660572.5
Time	9	27796	1770	9.93	9.54	658045.9		
Unit	10	1447	577	3.18	0.11	547678.8		
ResD	10	3402	597	15.06	0.44	547678.8	560405.9	-
Time	10	28462	1827	9.92	9.58	560099.7		

Table 8.9: Column generation – MUCP instances – (n,T) = (20,48), 2-peak per day demand

	id	#iter	#col	CPU	M-CPU	Dual b.	CplexCuts	Opt
Unit	1	2584	708	4.79	0.39	656378.2		
Site	1	3581	639	17.78	0.56	656378.3	670197	-
ResD	1	3178	618	19.06	0.35	656378.3		
Time	1	25315	1601	8.52	8.33	671006		
Unit	2	2333	666	3.72	0.26	656652.1		
Site	2	2438	554	11.51	0.38	656652.1	666635.6	-
ResD	2	2654	579	15.24	0.3	656652.1		
Time	2	29039	1796	12.22	11.91	668268.9		
Unit	3	1295	489	2.78	0.18	504012.6		
Site	3	2261	468	11.14	0.25	504012.6	516629	523523.7
ResD	3	1860	431	12.28	0.22	504012.6		
Time	3	38720	2160	15.38	15.04	517075.2		
Unit	4	1712	592	3.79	0.23	652805.5		
Site	4	2680	584	15.53	0.36	652805.5	664403.9	-
ResD	4	2746	557	15.87	0.39	652805.5		
Time	4	34122	1986	13.09	12.83	665189		
Unit	5	2083	622	3.68	0.24	687085		
Site	5	5184	734	27.8	0.68	687091.7	697794.4	-
ResD	5	4675	688	37.75	0.68	687091.7		
Time	5	25184	1644	8.6	8.3	698827.5		
Unit	6	1351	524	3.14	0.23	542302.5		
Site	6	2115	495	9.87	0.25	542302.5	552669.8	-
ResD	6	1870	445	9.86	0.26	542302.5		
Time	6	41948	2222	17.66	17.27	553417.1		
Unit	7	1979	611	3.56	0.26	632707.7		
Site	7	2559	515	9.24	0.27	632707.7	641479.5	-
ResD	7	2661	523	10.96	0.26	632707.7		
Time	7	35564	1845	13.06	12.69	642214.3		
Unit	8	1792	564	3.2	0.19	629944.8		
Site	8	2508	542	11.14	0.29	629944.8	641912.2	-
ResD	8	2741	562	14.48	0.29	629944.8		
Time	8	32853	1692	12.27	11.9	642586.2		
Unit	9	2353	627	3.91	0.31	646767.6		
Site	9	3662	629	19.81	0.51	646767.6	657268.7	660572.5
ResD	9	3930	608	23.82	0.57	646767.6		
Time	9	26900	1737	10.54	10.21	658088.8		
Unit	10	1878	560	2.93	0.18	548523.6		
Site	10	2610	506	11.46	0.4	548523.6	560593	-
ResD	10	2856	536	12.52	0.33	548523.6		
Time	10	25690	1786	9.84	9.52	560434		

Table 8.10: Column generation – IMUCP instances – (n,T) = (20,48), 2-peak per day demand

	id	#iter	#col	CPU	M-CPU	Dual b.	CplexCuts	Opt
Unit	1	223	245	0.76	0.02	392323.4		
ResD	1	368	178	2.07	0.03	393063	397350.2	398921.9
Time	1	18620	1229	5.29	5.03	397665		
Unit	2	215	241	0.74	0.02	332477.1		
ResD	2	276	128	3.32	0.02	332488.8	335765.1	337379.4
Time	2	24380	1601	8.29	8.03	336216.4		
Unit	3	181	237	0.97	0.02	482514.6		
ResD	3	89	58	0.66	0.03	491217.3	498993.6	498993.6
Time	3	46275	2486	13.07	12.46	498993.6		
Unit	4	214	196	0.56	0.05	236583.8		
ResD	4	296	134	3.18	0.02	236583.8	239499	241334.5
Time	4	19710	1486	6.62	6.32	239852		
Unit	5	160	226	0.95	0.02	469952.5		
ResD	5	341	147	2.35	0.02	470236.3	475751.5	477291
Time	5	43185	2659	19.26	18.5	476367.5		
Unit	6	165	206	0.81	0.04	461897.4		
ResD	6	205	112	1.75	0.04	461900.5	467454.2	469694.5
Time	6	32749	2149	16.4	15.86	468159.4		
Unit	7	154	209	0.54	0.02	421607.9		
ResD	7	180	104	0.91	0.02	425131.4	430967.6	430967.6
Time	7	604942	6369	250.32	243.84	430967.6		
Unit	8	200	219	0.58	0.05	480003.5		
ResD	8	245	109	2.55	0.01	482066	488776.3	488810.2
Time	8	22115	1731	7.44	7.11	488420.5		
Unit	9	247	206	0.84	0.02	344502.1		
ResD	9	414	168	2.64	0.02	346989.5	350689.9	353102.3
Time	9	21556	1649	8.05	7.71	351042.5		
Unit	10	158	252	0.63	0.03	472490.1		
ResD	10	131	88	0.66	0.02	478822	486431.6	486470.2
Time	10	87091	3537	27.67	26.79	485981.5		

Table 8.11: Column generation – MUCP instances – (n,T) = (20,24), random demand

	id	#iter	#col	CPU	M-CPU	Dual b.	CplexCuts	Opt
Unit	1	436	278	1.26	0.03	393972.3		
Site	1	427	199	2.35	0.07	393973.1	399348.6	399380.3
ResD	1	364	156	2.34	0.01	394139.9		
Time	1	19763	1284	5.9	5.74	398924.7		
Unit	2	412	276	0.88	0.04	337706.4		
Site	2	265	136	1.01	0.01	337706.4	340218.9	342396.8
ResD	2	237	111	1.9	0.02	337724.7		
Time	2	23497	1487	7.7	7.42	340423.7		
Unit	3	259	241	1.16	0.04	494664.3		
Site	3	183	108	0.98	0.01	494664.3	508412.1	508412.1
ResD	3	80	51	0.73	0.02	500577.4		
Time	3	39815	2504	13.54	13.03	507850.9		
Unit	4	399	227	0.64	0.02	239504		
Site	4	235	123	1.23	0.02	239504	242010.4	244191.5
ResD	4	225	123	1.7	0.02	239504		
Time	4	21471	1486	6.94	6.75	242438.5		
Unit	5	278	240	0.98	0.02	471883.5		
Site	5	282	149	1.97	0.04	471883.5	478634.8	480528.4
ResD	5	277	105	1.46	0.03	472236.7		
Time	5	34095	2146	11.88	11.46	478931.6		
Unit	6	249	248	0.94	0.04	468353.4		
Site	6	208	127	1.77	0.03	468353.4	475162.8	475770.3
ResD	6	187	114	1.89	0.01	468412.7		
Time	6	28364	1887	13.13	12.68	474241.1		
Unit	7	289	225	0.72	0.06	426422.4		
Site	7	275	151	1	0.03	426423.2	437726.4	437749.5
ResD	7	167	92	1.17	0.02	430134.7		
Time	7	535048	4362	162.67	159.24	436987.4		
Unit	8	414	243	0.98	0.02	491547.2		
Site	8	297	137	2.8	0.02	491648.9	500418.3	500451.7
ResD	8	240	106	2.64	0.02	492927.5		
Time	8	19517	1826	7.66	7.32	499999.2		
Unit	9	405	213	0.88	0.03	345786.2		
Site	9	380	156	2.39	0.04	345786.2	351411.9	354172.7
ResD	9	368	153	1.81	0.03	348252.5		
Time	9	18698	1503	6.02	5.72	352026.1		
Unit	10	291	227	0.8	0.02	480402.5		
Site	10	193	149	0.92	0.03	480402.5	494819.7	494865.6
ResD	10	120	85	1.18	0.01	487557.7		
Time	10	61304	3032	21.55	20.9	493896.3		

Table 8.12: Column generation – IMUCP instances – (n,T) = (20,24), random demand

	id	#iter	#col	CPU	M-CPU	Dual b.	CplexCuts	Opt
Unit	1	487	405	2.78	0.06	920857.1		
ResD	1	564	195	5.53	0.08	930400	939303.7	943533.2
Time	1	177900	5812	121.81	119.62	939571.1		
Unit	2	495	468	1.97	0.06	686193.5		
ResD	2	828	265	10.63	0.08	687232.5	697399.1	698487.1
Time	2	56254	3544	45.27	44.04	697771.7		
Unit	3	500	439	3.06	0.06	800431.5		
ResD	3	566	207	3.13	0.1	804487.1	812477.8	815818
Time	3	140210	5700	74.99	73.12	812821.1		
Unit	4	634	463	3.72	0.09	780467.7		
ResD	4	991	317	14.44	0.13	782872.9	789211	792735.8
Time	4	44040	2911	29	28.47	790206.1		
Unit	5	420	410	2	0.05	740714.2		
ResD	5	624	187	11.18	0.1	748378.8	758597.2	760119.8
Time	5	72217	4064	61.39	59.63	758032.1		
Unit	6	417	383	1.97	0.1	782174.7		
ResD	6	631	243	5.98	0.16	785400.1	795273.5	796467.4
Time	6	380025	7861	229.24	223.59	795130.6		
Unit	7	441	392	2.39	0.07	823020.2		
ResD	7	568	207	8.95	0.1	824901.3	835744.8	838005
Time	7	67300	3929	44.14	43.13	836434.7		
Unit	8	440	417	2.03	0.08	844069.1		
ResD	8	464	185	9.92	0.1	848825.8	858709.1	858989.9
Time	8	136659	5939	85.04	82.9	857575.4		
Unit	9	513	444	2.01	0.05	867092.7		
ResD	9	633	212	10.2	0.06	870737.9	883881.8	886175.7
Time	9	64903	3966	50.9	49.73	883507.3		
Unit	10	539	403	1.97	0.07	758405.7		
ResD	10	759	257	11.46	0.1	764246.9	774068.4	775035.8
Time	10	698841	8854	431.9	426.21	772901.4		

Table 8.13: Column generation – MUCP instances – (n,T) = (20,48), random demand

	id	#iter	#col	CPU	M-CPU	Dual b.	CplexCuts	Opt
Unit	1	1001	483	4.5	0.18	925700.6		
Site	1	650	267	6.53	0.21	925700.6	943238.1	947499.3
ResD	1	473	168	4.64	0.05	934913.9		
Time	1	171006	5329	125.46	123.28	943918.5		
Unit	2	1479	562	3.22	0.19	695401.1		
Site	2	665	300	7.38	0.12	695527.4	705573.9	707006.7
ResD	2	657	258	11.9	0.07	695808.2		
Time	2	50759	3469	48.46	47.38	705621.5		
Unit	3	1101	479	3.96	0.14	806978.7		
Site	3	638	325	6.97	0.11	807203.1	820524.3	823628.3
ResD	3	491	222	3.69	0.1	811269.6		
Time	3	161271	5398	87.62	85.66	819637.3		
Unit	4	1511	527	4.66	0.15	783981.4		
Site	4	1053	352	8.68	0.09	783981.4	792312.7	796652.9
ResD	4	978	320	14.44	0.16	785978.5		
Time	4	46782	2919	33.3	32.69	793075.6		
Unit	5	1318	536	3.6	0.12	761059		
Site	5	772	256	6.37	0.09	761075.1	775091.7	776601
ResD	5	587	203	11.09	0.07	765707.9		
Time	5	61708	3772	47.92	46.83	774724.1		
Unit	6	708	413	2.03	0.1	786073.7		
Site	6	747	313	4.46	0.11	786073.7	799731.7	801678.7
ResD	6	564	244	5.93	0.07	790482.9		
Time	6	344097	8685	276.9	270.05	799759.4		
Unit	7	1011	470	2.95	0.13	838014		
Site	7	689	317	7.56	0.09	838054.2	848575.8	850659.3
ResD	7	479	196	9.38	0.07	839828.8		
Time	7	57993	3787	41.3	40.45	849302.1		
Unit	8	1199	486	2.8	0.13	866885.7		
Site	8	658	251	7.08	0.14	866898.1	880311.4	882188.9
ResD	8	447	189	13.24	0.1	871522.5		
Time	8	122579	5314	96.04	93.53	880744.1		
Unit	9	1331	537	3.82	0.22	891460.1		
Site	9	858	314	8.05	0.07	891517.1	901564.1	907044.6
ResD	9	533	206	10.43	0.09	892802.8		
Time	9	58913	3645	41.36	40.6	900954.5		
Unit	10	1244	495	2.42	0.13	774815.2		
Site	10	732	297	4.26	0.11	774815.2	790185.1	790241.9
ResD	10	572	216	9.88	0.1	779617.2		
Time	10	891113	9547	634.47	625.65	789100.1		

Table 8.14: Column generation – IMUCP instances – (n,T) = (20,48), random demand

Column generation results for start-up decompositions Table 8.15 presents the results of Unit-SU, the start-up decomposition (implemented within a unit decomposition structure), compared to Unit on 2-peak-demand instances.

Table 8.16 presents the results of Site-SU, the start-up decomposition (implemented within a site decomposition structure) compared to Site for random-demand instances.

For both tables, the column entries are the same as in Tables 8.7 to 8.14.

		#iter	#col	CPU	M-CPU	Dual b.
T = 24	Unit	341	256	0.87	0.03	321704.4
	Unit-SU	2410	248	1.29	0.13	321704.4
	Unit	459	295	0.94	0.04	308326.5
	Unit-SU	2911	224	1.24	0.16	308326.5
	Unit	440	257	0.73	0.01	231853.3
	Unit-SU	2632	250	1.38	0.16	231853.3
	Unit	443	269	0.91	0.04	301511
	Unit-SU	2594	256	1.41	0.17	301511
	Unit	557	324	0.96	0.04	247949.3
	Unit-SU	3120	265	1.3	0.15	247949.3
	Unit	480	286	0.85	0.04	302446
	Unit-SU	2642	226	1.19	0.11	302446
	Unit	417	255	0.74	0.03	273110.7
	Unit-SU	2704	238	1.22	0.18	273110.7
	Unit	514	298	1.02	0.04	311345.7
	Unit-SU	2590	247	1.28	0.15	311345.7
	Unit	469	273	0.8	0.05	255163.7
	Unit-SU	2456	233	1.24	0.16	255163.7
	Unit	528	287	0.92	0.07	336067
	Unit-SU	3017	247	1.4	0.22	336067
T = 48	Unit	2584	708	5.01	0.37	656378.2
	Unit-SU	8789	493	5.6	1.43	656378.2
	Unit	2333	666	3.82	0.31	656652.1
	Unit-SU	8393	445	4.41	1.3	656652.1
	Unit	1295	489	2.63	0.14	504012.6
	Unit-SU	6876	390	4.13	1.05	504012.6
	Unit	1712	592	3.8	0.24	652805.5
	Unit-SU	7709	411	4.51	1.23	652805.5
	Unit	2083	622	3.66	0.29	687085
	Unit-SU	8540	459	5.02	1.45	687085
	Unit	1351	524	3.21	0.2	542302.5
	Unit-SU	6782	427	4.63	1.07	542302.5
	Unit	1979	611	3.62	0.25	632707.7
	Unit-SU	8049	414	4.74	1.28	632707.7
	Unit	1792	564	3.28	0.29	629944.8
	Unit-SU	8454	427	4.5	1.18	629944.8
	Unit	2353	627	4.01	0.34	646767.6
	Unit-SU	8443	492	5.45	1.47	646767.6
	Unit	1878	560	2.96	0.22	548523.6
	Unit-SU	8418	458	4.48	1.34	548523.6

Table 8.15: Column generation for start-up decomposition – IMUCP instances with 2-peak per day demand

		#iter	#col	CPU	M-CPU	Dual b.
T = 24	Site	427	199	2.48	0.03	393973.1
	Site-SU	2075	136	2.67	0.18	393972.3
	Site	265	136	1.13	0.03	337706.4
	Site-SU	2133	98	1.74	0.15	337706.4
	Site	183	108	1.06	0.03	494664.3
	Site-SU	1370	77	0.86	0.1	494664.3
	Site	235	123	1.3	0.02	239504
	Site-SU	2109	77	0.99	0.15	239504
	Site	282	149	2.07	0.04	471883.5
	Site-SU	1547	99	1.16	0.12	471883.5
	Site	208	127	1.84	0.01	468353.4
	Site-SU	2077	104	1.36	0.16	468353.4
	Site	275	151	1.77	0.03	426423.2
	Site-SU	2266	152	3.84	0.12	426422.4
	Site	297	137	3.86	0.01	491648.9
	Site-SU	1550	93	2.48	0.16	491547.2
	Site	380	156	2.53	0.05	345786.2
	Site-SU	2308	119	1.99	0.23	345786.2
	Site	193	149	1.29	0.03	480402.5
	Site-SU	1393	110	1.9	0.15	480402.5
T = 48	Site	650	267	6.85	0.11	925700.6
	Site-SU	3940	195	5.72	0.65	925700.6
	Site	665	300	7.03	0.09	695527.4
	Site-SU	4728	187	9.2	0.56	695401.1
	Site	638	325	6.73	0.15	807203.1
	Site-SU	4899	254	7.8	0.68	807029.3
	Site	1053	352	8.94	0.15	783981.4
	Site-SU	6059	227	7.09	1.01	783981.4
	Site	772	256	6.64	0.08	761075.1
	Site-SU	5148	197	7.01	0.82	761075.1
	Site	747	313	4.48	0.07	786073.7
	Site-SU	4945	207	4.54	0.63	786073.7
	Site	689	317	7.54	0.15	838054.2
	Site-SU	4739	199	8.15	0.71	838014
	Site	658	251	6.24	0.13	866898.1
	Site-SU	5991	303	21.26	1.95	866898.1
	Site	858	314	9.1	0.16	891517.1
	Site-SU	5421	291	26.03	1.73	891462.6
	Site	732	297	4.41	0.15	774815.2
	Site-SU	4601	183	5.71	0.69	774815.2

Table 8.16: Column generation for start-up decomposition – IMUCP instances with random demand

Column generation results for time decomposition with interval up-set inequalities
For MUCP and IMUCP instances, we compare Time and Time+I, the time decomposition with separation of interval up-set inequalities.

Table 8.17 (resp. 8.18) presents the results for $(n,T) = (20,48)$ MUCP and IMUCP instances featuring 2-peak-per-day (resp. random) demand. The column entries are the same as in Table 8.7, with additional entry

#IUP	the total number of interval up-set inequalities added.

	id	#IUP	#iter	#col	CPU	M-CPU	Dual b.
MUCP							
Time	1	-	24375	1569	8.55	8.32	670940.8
Time+I	1	2	24378	1569	8.33	8.13	670940.9
Time	2	-	30053	1801	13.47	13.13	668217.4
Time+I	2	0	30053	1801	13.19	12.87	668217.4
Time	3	-	34952	2120	15.04	14.61	516953.5
Time+I	3	1	35044	2123	15.29	14.94	516957.3
Time	4	-	33468	1834	12.25	11.94	665131.6
Time+I	4	3	33595	1839	12.88	12.49	665140.4
Time	5	-	29300	1649	9.21	9.03	698809.5
Time+I	5	3	29399	1654	9.69	9.5	698828.3
Time	6	-	38587	2188	16.6	16.18	553409.6
Time+I	6	6	38784	2202	16.7	16.19	553430.3
Time	7	-	33713	1863	13.54	13.2	642197.1
Time+I	7	5	34081	1891	15.08	14.65	642231.3
Time	8	-	26037	1735	9.64	9.28	642386.8
Time+I	8	2	26039	1735	9.5	9.16	642386.8
Time	9	-	27796	1770	10.38	10.17	658045.9
Time+I	9	0	27796	1770	10.57	10.22	658045.9
Time	10	-	28462	1827	10.7	10.31	560099.7
Time+I	10	1	28476	1830	11.04	10.63	560102.9
IMUCP							
Time	1	-	25315	1601	8.72	8.57	671006
Time+I	1	0	25315	1601	8.69	8.47	671006
Time	2	-	29039	1796	12.32	12.02	668268.9
Time+I	2	0	29039	1796	12.31	12.07	668268.9
Time	3	-	38720	2160	15.86	15.5	517075.2
Time+I	3	1	38765	2164	16.45	16.04	517086.5
Time	4	-	34122	1986	13.38	13.14	665189
Time+I	4	3	34190	1988	13.62	13.29	665195.1
Time	5	-	25184	1644	8.99	8.77	698827.5
Time+I	5	1	25311	1646	9.1	8.83	698839.5
Time	6	-	41948	2222	17.99	17.58	553417.1
Time+I	6	5	42081	2225	18.26	17.83	553436.3
Time	7	-	35564	1845	13.42	13.13	642214.3
Time+I	7	5	35843	1858	14.56	14.15	642247
Time	8	-	32853	1692	12.53	12.23	642586.2
Time+I	8	0	32853	1692	12.56	12.22	642586.2
Time	9	-	26900	1737	10.78	10.48	658088.8
Time+I	9	0	26900	1737	10.91	10.67	658088.8
Time	10	-	25690	1786	10.31	10.1	560434
Time+I	10	1	25746	1787	10.31	10.08	560438.6

Table 8.17: Column generation for time decomposition with interval up-set inequalities – (I)MUCP instances – $(n, T) = (20, 48)$ and 2-peak per day demand

	id	#IUP	#iter	#col	CPU	M-CPU	Dual b.
MUCP							
Time	1	-	177900	5812	129.54	127.47	939571.1
Time+I	1	0	177900	5812	128.28	126.25	939571.1
Time	2	-	56254	3544	43.96	42.8	697771.7
Time+I	2	0	56254	3544	43.78	42.7	697771.7
Time	3	-	140210	5700	76.89	75.27	812821.1
Time+I	3	0	140210	5700	76.71	74.99	812821.1
Time	4	-	44040	2911	28.84	28.37	790206.1
Time+I	4	0	44040	2911	29.13	28.52	790206.1
Time	5	-	72217	4064	62.29	60.41	758032.1
Time+I	5	1	74248	4227	71.27	69.1	758229.1
Time	6	-	380025	7861	235.6	229.71	795130.6
Time+I	6	2	381449	7904	237.2	231.68	795252
Time	7	-	67300	3929	46.42	45.45	836434.7
Time+I	7	0	67300	3929	47.17	46.16	836434.7
Time	8	-	136659	5939	91.12	88.97	857575.4
Time+I	8	0	136659	5939	92.14	89.69	857575.4
Time	9	-	64903	3966	51.82	50.67	883507.3
Time+I	9	0	64903	3966	51.52	50.57	883507.3
Time	10	-	698841	8854	451.88	446.13	772901.4
Time+I	10	2	698918	8855	450.43	445.08	772936.1
IMUCP							
Time	1	-	171006	5329	128.34	126.19	943918.5
Time+I	1	0	171006	5329	126.81	124.69	943918.5
Time	2	-	50759	3469	49.72	48.54	705621.5
Time+I	2	6	52097	3639	60.51	58.88	705670
Time	3	-	161271	5398	88.51	86.81	819637.3
Time+I	3	0	161271	5398	88.08	86.29	819637.3
Time	4	-	46782	2919	31.14	30.66	793075.6
Time+I	4	0	46782	2919	31.67	31.14	793075.6
Time	5	-	61708	3772	48.98	47.96	774724.1
Time+I	5	15	64325	3955	74.31	72.62	775196.7
Time	6	-	344097	8685	278.68	272.14	799759.4
Time+I	6	4	345117	8959	330.82	322.88	800349.1
Time	7	-	57993	3787	41.58	40.74	849302.1
Time+I	7	0	57993	3787	41.22	40.33	849302.1
Time	8	-	122579	5314	99.1	96.71	880744.1
Time+I	8	0	122579	5314	98.33	95.93	880744.1
Time	9	-	58913	3645	42.58	41.72	900954.5
Time+I	9	0	58913	3645	42.7	41.84	900954.5
Time	10	-	891113	9547	654.57	646.34	789100.1
Time+I	10	0	891113	9547	668.26	659.85	789100.1

Table 8.18: Column generation for time decomposition with interval up-set inequalities – (I)MUCP instances – $(n, T) = (20, 48)$ and random demand

Branch & Price results on small instances Tables 8.19 to 8.22 Branch & Price results on small-size, $i.e.$, $(n,T) = (20,24)$ instances. They present, for each instance and each decomposition structure:

id	the instance number,
#nodes	the number of nodes,
IUP	the number of interval up-set inequalities separated
#col	the number of columns generated,
CPU	the CPU time (in seconds) of the Branch & Price,
Gap	the optimality gap
Primal b.	the best integer solution found within the time limit

	id	nodes	IUP	#iter	#col	CPU	Gap	Primal b.
Unit	1	103338	-	1867177	46540717	3600	0.06917833	197355.2
Time	1	1821	-	272294	68994	41.06	0	195336.8
Time+I	1	1088	107	194857	38609	28.49	0	195336.8
Unit	2	99747	-	1771135	25385147	3600	0.05655635	157597.9
Time	2	15	-	3594	375	0.37	0	157498.7
Time+I	2	13	5	3942	465	0.43	0	157498.7
Unit	3	106391	-	1586302	29267900	3600	0.06795943	160891.5
Time	3	189	-	29003	5857	3.3	0	159635
Time+I	3	177	32	31489	4910	3.68	0	159635
Unit	4	117955	-	1689559	14017540	3600.01	0.04041181	183335
Time	4	9	-	3195	458	0.38	0	183335
Time+I	4	11	15	4323	481	0.54	0	183335
Unit	5	104485	-	1621957	14398496	3600.01	0.07047237	158036.4
Time	5	412	-	45626	11637	7.5	0	156820.5
Time+I	5	349	45	50945	9423	8.1	0	156820.5
Unit	6	118568	-	1807875	35046798	3600	0.05146701	182644.4
Time	6	23	-	4469	523	0.56	0	181659.2
Time+I	6	17	7	4088	529	0.53	0	181659.2
Unit	7	116043	-	1724905	19366839	3600.01	0.04716928	114095.3
Time	7	63	-	10556	1047	1	0	114095.3
Time+I	7	31	16	5796	593	0.55	0	114095.3
Unit	8	99145	-	2051269	32343718	3600	0.04937396	161224.6
Time	8	143	-	18603	2493	2.06	0	161224.6
Time+I	8	135	14	17397	1915	2.07	0	161224.6
Unit	9	116138	-	1678076	6044792	3600	0.05723049	153050
Time	9	97	-	25363	3778	2.9	0	152396.6
Time+I	9	67	20	19286	2241	2.76	0	152396.6
Unit	10	117788	-	1640528	5188730	3600.01	0.03963	151060.9
Time	10	9	-	3016	413	0.31	0	151060.9
Time+I	10	9	0	3016	413	0.32	0	151060.9

Table 8.19: Branch & Price – MUCP instances – $(n,T) = (10,24)$ and 2-peak demand

	id	nodes	IUP	#iter	#col	CPU	Gap	Primal b.
Unit	1	102980	-	2045862	41462167	3600	0.06472579	196441.5
Time	1	1334	-	214586	45416	30.89	0	195336.8
Time+I	1	900	83	163043	31847	23.51	0	195336.8
Unit	2	96012	-	1893608	35042602	3600	0.06229561	158665
Time	2	15	-	4497	435	0.44	0	157498.7
Time+I	2	13	5	3596	405	0.39	0	157498.7
Unit	3	105177	-	1802987	38027145	3600	0.0788929	162469.1
Time	3	214	-	32230	8341	3.88	0	159635
Time+I	3	204	30	35288	6554	4.74	0	159635
Unit	4	116954	-	1751754	13479810	3600.01	0.04058776	183335
Time	4	9	-	2962	448	0.37	0	183335
Time+I	4	11	15	3839	470	0.51	0	183335
Unit	5	104105	-	1762066	15057665	3600.01	0.07176651	158338.9
Time	5	343	-	47055	10388	7.21	0	156820.5
Time+I	5	427	39	66825	13873	10.03	0	156820.5
Unit	6	116725	-	2011594	31678699	3600	0.05233168	182506.6
Time	6	21	-	6584	752	0.76	0	181659.2
Time+I	6	17	8	4800	601	0.7	0	181659.2
Unit	7	118535	-	1786025	14589643	3600	0.06087007	115124.6
Time	7	62	-	11330	1000	1.07	0	114095.3
Time+I	7	37	17	7062	1239	0.8	0	114095.3
Unit	8	94953	-	2393061	28427989	3600.02	0.06103055	162797.5
Time	8	130	-	19359	2575	2.1	0	161224.6
Time+I	8	101	23	17114	1536	1.96	0	161224.6
Unit	9	115248	-	1687646	5305873	3600.01	0.06136625	153653.7
Time	9	79	-	21585	3148	2.58	0	152396.6
Time+I	9	51	13	18761	2201	2.26	0	152396.6
Unit	10	115824	-	1738714	5690337	3600.01	0.03906384	151060.9
Time	10	7	-	2515	375	0.36	0	151060.9
Time+I	10	7	0	2515	375	0.31	0	151060.9

Table 8.20: Branch & Price – IMUCP instances – $(n,T) = (10,24)$ and 2-peak-per-day demand

ResD	1	21686	-	407848	41391250	3600.03	0.02943064	164351.4
Time	1	11	-	8655	1488	1.39	0	164351.4
Time+I	1	21	2	10960	1931	2.09	0	164351.4
ResD	2	31210	-	432225	120651200	3600.09	0.009958067	215933.8
Time	2	5	-	7411	691	0.9	0	215933.8
Time+I	2	5	0	7411	691	0.92	0	215933.8
ResD	3	11659	-	112253	48459698	2484.92	0	193044.1
Time	3	1	-	3097	438	0.24	0	193044.1
Time+I	3	1	0	3097	438	0.23	0	193044.1
ResD	4	70567	-	778175	240561029	3600	0.01017191	206617.5
Time	4	19	-	3104	812	0.62	0	206617.5
Time+I	4	17	1	2953	766	0.61	0	206617.5
ResD	5	72871	-	725049	153101235	3600	0.01656212	238019
Time	5	39	-	14128	3372	2.53	0	238019
Time+I	5	27	7	10618	1936	1.87	0	238019
ResD	6	30058	-	333446	248178355	3600	0.01918268	194335.9
Time	6	3	-	4495	471	0.34	0	194335.9
Time+I	6	3	0	4495	471	0.34	0	194335.9
ResD	7	25587	-	399593	117784312	3600	0.007877554	229356.4
Time	7	3	-	10350	709	0.95	0	229356.4
Time+I	7	3	0	10350	709	0.94	0	229356.4
ResD	8	28627	-	267520	53247723	3600.01	0.0006243973	235930.3
Time	8	1	-	4411	472	0.35	0	235930.3
Time+I	8	1	0	4411	472	0.36	0	235930.3
ResD	9	48454	-	899413	100087292	3600	0.02120061	201594.4
Time	9	1	-	4559	519	0.45	0	201594.4
Time+I	9	1	0	4559	519	0.46	0	201594.4
ResD	10	24470	-	396810	85814320	3600.01	0.01258229	207772.7
Time	10	9	-	8675	685	1.13	0	207772.7
Time+I	10	9	1	10025	736	1.28	0	207772.7

Table 8.21: Branch & Price – MUCP instances – $(n,T) = (10,24)$ and random demand

	id	nodes	IUP	#iter	#col	CPU	Gap	Primal b.
ResD	1	25843	-	445347	44692611	3600.01	0.02055399	166475.7
Time	1	13	-	7430	922	1.1	0	166475.7
Time+I	1	9	2	6722	704	0.85	0	166475.7
ResD	2	35596	-	497482	94044926	3600.06	0.008248965	215933.8
Time	2	3	-	6658	578	0.67	0	215933.8
Time+I	2	3	0	6658	578	0.67	0	215933.8
ResD	3	9221	-	83008	27240446	1505.35	0	193909
Time	3	1	-	4417	532	0.45	0	193909
Time+I	3	1	0	4417	532	0.49	0	193909
ResD	4	47581	-	561335	90289194	3600.01	0.01350733	208933.9
Time	4	11	-	3596	562	0.55	0	208933.9
Time+I	4	11	0	3596	562	0.57	0	208933.9
ResD	5	73285	-	785562	115731273	3600	0.01792698	239791.8
Time	5	55	-	13651	2892	2.54	0	239791.8
Time+I	5	31	3	10751	1685	2.18	0	239791.8
ResD	6	29935	-	355242	168136986	3600	0.0094393	195989.4
Time	6	1	-	3268	446	0.26	0	195989.4
Time+I	6	1	0	3268	446	0.27	0	195989.4
ResD	7	21975	-	263282	61069768	2902.24	0	230614.8
Time	7	3	-	9099	617	0.92	0	230614.8
Time+I	7	3	0	9099	617	0.91	0	230614.8
ResD	8	33051	-	323716	62452293	2594.71	0	237541.9
Time	8	1	-	4043	473	0.32	0	237541.9
Time+I	8	1	0	4043	473	0.33	0	237541.9
ResD	9	42726	-	795751	74541598	3600.08	0.02259438	202446.1
Time	9	3	-	5655	558	0.65	0	202446.1
Time+I	9	3	0	5655	558	0.63	0	202446.1
ResD	10	21025	-	410812	71084450	3600.01	0.01042285	212188.9
Time	10	9	-	10016	1121	1.78	0	212188.9
Time+I	10	11	2	9348	1068	1.84	0	212188.9

Table 8.22: Branch & Price – IMUCP instances – $(n,T) = (10,24)$ and random demand

Branch & Price results on large instances Tables 8.23 (resp. 8.24) present the results for $(20,24)$ and $(20,48)$ 2-peak-demand MUCP (resp. IMUCP) instances. Tables 8.25 (resp. 8.26) present the results for $(20,24)$ and $(20,48)$ random-demand MUCP (resp. IMUCP) instances. The column entries are the same as Table 8.19.

	id	nodes	IUP	#iter	#col	CPU	Gap	Primal b.
$(n,T) = (20,24)$								
Time	1	3511	-	1955295	1419867	1372	0 %	332471.5
Time+I	1	4649	19	2660287	1920532	1854	0 %	332471.5
Time	2	1854	-	770417	186761	274	0 %	316585.4
Time+I	2	2143	13	897501	219175	334	0 %	316585.4
Time	3	3841	-	1086638	466607	486	0 %	242467.1
Time+I	3	1644	18	528000	164381	246	0 %	242467.1
Time	4	9665	-	3566367	1550248	1472	0 %	311586.1
Time+I	4	3501	88	1404115	559602	740	0 %	311586.1
Time	5	34354	-	6459551	2381886	2399	0 %	254636
Time+I	5	33417	68	6439876	2428150	2908	0 %	254636
Time	6	103	-	79345	8735	31	0 %	309777.4
Time+I	6	104	3	86315	7865	33	0 %	309777.4
Time	7	11680	-	3328242	1123932	1182	0 %	280374.3
Time+I	7	12129	78	3597963	1253193	1441	0 %	280374.3
Time	8	3464	-	1067308	266283	389	0 %	319807.6
Time+I	8	2411	23	749304	191788	312	0 %	319807.6
Time	9	666	-	186764	47428	71	0 %	263334.5
Time+I	9	669	7	206035	48318	78	0 %	263334.5
Time	10	4879	-	1607504	546965	591	0 %	344357.9
Time+I	10	4716	8	1527203	520075	605	0 %	344357.9
$(n,T) = (20,48)$								
Time	1	6395	-	3066051	658694	3600	2.4 %	688233.5
Time+I	1	6066	86	2916982	624524	3600	2.4 %	688233.5
Time	2	3121	-	2422588	564995	3600	1.8 %	681553.9
Time+I	2	2986	37	2303078	535149	3600	1.8 %	681553.9
Time	3	5622	-	2821403	1203629	3600	1.1 %	524859.4
Time+I	3	4516	111	2661982	941533	3600	1.1 %	524859.4
Time	4	3690	-	2606505	559369	3600	1.5 %	676863
Time+I	4	3291	102	2293518	494279	3600	1.5 %	676863
Time	5	6810	-	3147406	1011539	3600	0.7 %	704615.5
Time+I	5	5657	57	2890395	847174	3600	0.6 %	704615.5
Time	6	3526	-	2682858	851744	3600	1.6 %	564739.3
Time+I	6	3146	140	2344402	739096	3600	1.6 %	564739.3
Time	7	3309	-	2460602	659801	3600	1.6 %	653771
Time+I	7	3067	78	2379993	586708	3600	1.6 %	653771
Time	8	9029	-	3216025	815769	3600	1.5 %	653702.4
Time+I	8	8091	74	2931292	696495	3600	1.5 %	653702.4
Time	9	6276	-	3162322	729402	3600	0.8 %	664261.3
Time+I	9	5716	85	3085675	666247	3600	0.8 %	664261.3
Time	10	10530	-	3315132	1024260	3600	1.6 %	570713.8
Time+I	10	9504	37	3067064	905015	3600	1.7 %	570713.8

Table 8.23: Branch & Price – MUCP instances – 2-peak demand

	id	nodes	IUP	#iter	#col	CPU	Gap	Primal b.
$(n,T) = (20,24)$								
Time	1	2696	-	1570582	943438	1135	0 %	332471.5
Time+I	1	3146	16	1792827	1096515	1334	0 %	332471.5
Time	2	2556	-	985660	262211	428	0 %	316859.1
Time+I	2	2633	18	1027781	277425	469	0 %	316859.1
Time	3	6437	-	2064878	840889	861	0 %	242612.1
Time+I	3	4013	33	1276283	578423	628	0 %	242612.1
Time	4	21290	-	8312914	3965216	3600	0 %	312102.7
Time+I	4	15158	209	6639811	2866182	3600	0.2 %	312346.6
Time	5	41127	-	8573229	3237565	3375	0 %	254675.6
Time+I	5	37231	68	8150422	2928347	3600	0 %	254675.6
Time	6	189	-	167065	24298	72	0 %	310647.6
Time+I	6	193	1	179127	24664	73	0 %	310647.6
Time	7	13195	-	3803894	1296179	1408	0 %	280374.3
Time+I	7	13629	72	4155665	1423896	1730	0 %	280374.3
Time	8	5919	-	1730164	549754	706	0 %	320128.2
Time+I	8	6305	36	1932071	588933	827	0 %	320128.2
Time	9	407	-	130778	24489	47	0 %	263334.5
Time+I	9	427	1	142294	28133	55	0 %	263334.5
Time	10	19120	-	6487588	2862315	2678	0 %	344836.8
Time+I	10	15630	12	5193516	2361722	2319	0 %	344836.8
$(n,T) = (20,48)$								
Time	1	6063	-	2686959	612565	3600	1.8 %	684241.3
Time+I	1	5540	87	2545088	545832	3600	1.8 %	684241.3
Time	2	2518	-	2023700	501599	3600	1.8 %	681634.4
Time+I	2	2453	43	2013081	481023	3600	1.8 %	681634.4
Time	3	4244	-	2464889	1027028	3600	1.6 %	527943.3
Time+I	3	4025	81	2535658	987966	3600	1.7 %	527943.3
Time	4	3397	-	2338676	493010	3600	2.4 %	682719.5
Time+I	4	3158	106	2217522	410209	3600	2.4 %	682719.5
Time	5	5773	-	2828086	846908	3600	1.1 %	707981.8
Time+I	5	5542	59	2719539	801179	3600	1.1 %	707981.8
Time	6	3341	-	2521538	824539	3600	1.4 %	563795.5
Time+I	6	3004	131	2307480	764993	3600	1.4 %	563795.5
Time	7	3022	-	2329888	627233	3600	1.7 %	654653.4
Time+I	7	2799	91	2174960	529566	3600	1.7 %	654653.4
Time	8	7507	-	3004539	703356	3600	1.5 %	653846.6
Time+I	8	6845	73	2718031	604058	3600	1.5 %	653846.6
Time	9	5189	-	2935061	611446	3600	0.8 %	664867.4
Time+I	9	4891	63	2855547	585821	3600	0.8 %	664867.4
Time	10	7191	-	2807633	837124	3600	1.6 %	571197.8
Time+I	10	6925	39	2607432	765795	3600	1.7 %	571197.8

Table 8.24: Branch & Price – IMUCP instances – 2-peak demand

	id	nodes	IUP	#iter	#col	CPU	Gap	Primal b.
$(n,T) = (20,24)$								
Time	1	969	-	604505	475183	363	0 %	398921.9
Time+I	1	1110	6	645269	453130	372	0 %	398921.9
Time	2	299	-	480640	231719	420	0 %	337379.4
Time+I	2	248	1	389450	184857	332	0 %	337379.4
Time	3	1	-	163893	3161	35	0 %	498993.6
Time+I	3	1	0	163893	3161	35	0 %	498993.6
Time	4	2131	-	2296661	2069115	3600	0.9 %	243302.5
Time+I	4	2121	0	2287509	2056965	3600	0.9 %	243302.5
Time	5	78	-	382448	56310	252	0 %	477291
Time+I	5	78	0	382448	56310	252	0 %	477291
Time	6	45	-	237523	43528	238	0 %	469694.5
Time+I	6	45	0	237523	43528	237	0 %	469694.5
Time	7	1	-	453216	6653	170	0 %	430967.6
Time+I	7	1	0	453216	6653	169	0 %	430967.6
Time	8	39	-	121256	11499	56	0 %	488810.2
Time+I	8	39	0	121256	11499	56	0 %	488810.2
Time	9	1647	-	2304244	2774871	3600	0.4 %	353960.7
Time+I	9	1589	5	2197528	2668511	3600	0.4 %	353960.7
Time	10	73	-	304199	45230	169	0 %	486470.2
Time+I	10	73	0	304199	45230	167	0 %	486470.2
$(n,T) = (20,48)$								
Time	1	406	-	3062852	329817	3600	0.7 %	947820.5
Time+I	1	406	0	3062453	325817	3600	0.7 %	947820.5
Time	2	242	-	1406302	316025	2394	0 %	698487.1
Time+I	2	171	12	441535	165538	947	0 %	698487.1
Time	3	1828	-	3386715	1035624	3600	0.2 %	816339.6
Time+I	3	1818	0	3367073	1027348	3600	0.2 %	816339.6
Time	4	1164	-	2283850	589557	3600	0.6 %	796033.9
Time+I	4	1098	2	2290894	580610	3600	0.6 %	796033.9
Time	5	69	-	395369	49032	470	0 %	760119.8
Time+I	5	137	22	620142	173259	990	0 %	760119.8
Time	6	421	-	1256729	552525	1742	0 %	796467.4
Time+I	6	91	2	521217	50236	463	0 %	796467.4
Time	7	272	-	1513241	349405	2193	0 %	838005
Time+I	7	265	1	1596381	320535	2245	0 %	838005
Time	8	411	-	3082869	522321	3600	0 %	858989.9
Time+I	8	375	12	3250347	435375	3600	0 %	858989.9
Time	9	373	-	3340336	358237	3600	0.5 %	889321.9
Time+I	9	373	0	3334005	358219	3600	0.5 %	889321.9
Time	10	1035	-	2595946	1404299	3250	0 %	775035.8
Time+I	10	511	2	872663	537575	1335	0 %	775035.8

Table 8.25: Branch & Price – MUCP instances – random demand

	id	nodes	IUP	#iter	#col	CPU	Gap	Primal b.
$(n,T) = (20,24)$								
Time	1	17	-	83385	3049	31	0 %	399380.3
Time+I	1	17	0	83385	3049	31	0 %	399380.3
Time	2	1025	-	1234416	1074272	1435	0 %	342396.8
Time+I	2	1031	5	1246388	1147520	1478	0 %	342396.8
Time	3	7	-	257588	5620	80	0 %	508412.1
Time+I	3	3	2	225191	4109	63	0 %	508412.1
Time	4	704	-	1066934	863417	1631	0 %	244191.5
Time+I	4	927	1	1391121	1135242	2328	0 %	244191.5
Time	5	213	-	569803	336861	667	0 %	480528.4
Time+I	5	213	0	569803	336861	667	0 %	480528.4
Time	6	69	-	334540	70180	313	0 %	475770.3
Time+I	6	69	0	334540	70180	312	0 %	475770.3
Time	7	7	-	420483	5734	163	0 %	437749.5
Time+I	7	7	0	420483	5734	163	0 %	437749.5
Time	8	47	-	142919	9053	54	0 %	500451.7
Time+I	8	47	0	142919	9053	54	0 %	500451.7
Time	9	1645	-	2330848	2298658	3600	0.1 %	354206.7
Time+I	9	1641	0	2324956	2285068	3600	0.1 %	354206.7
Time	10	123	-	336277	39084	205	0 %	494865.6
Time+I	10	123	0	336277	39084	205	0 %	494865.6
$(n,T) = (20,48)$								
Time	1	581	-	2551492	641958	3600	0.1 %	947499.3
Time+I	1	578	1	2540837	629737	3600	0.2 %	947499.3
Time	2	174	-	574718	265476	1585	0 %	707006.7
Time+I	2	143	17	628829	145212	1116	0 %	707006.7
Time	3	935	-	3519801	647550	3600	0.5 %	825066.5
Time+I	3	934	0	3510872	647242	3600	0.5 %	825066.5
Time	4	735	-	2036308	445857	3600	0.9 %	801509.4
Time+I	4	735	0	2034990	445791	3600	0.9 %	801509.4
Time	5	492	-	1956086	843576	3600	0 %	776601
Time+I	5	475	36	2261644	752790	3600	0 %	776601
Time	6	789	-	2053892	961514	2819	0 %	801678.7
Time+I	6	329	5	850754	254945	1234	0 %	801678.7
Time	7	240	-	2014025	511107	2901	0 %	850659.3
Time+I	7	317	7	2480122	681301	3600	0 %	850659.3
Time	8	61	-	572546	62402	746	0 %	882188.9
Time+I	8	61	0	572546	62402	745	0 %	882188.9
Time	9	476	-	2938952	325017	3600	0.9 %	910640.8
Time+I	9	501	1	2817490	351344	3600	0.9 %	910640.8
Time	10	49	-	526254	21204	403	0 %	790241.9
Time+I	10	55	1	543818	25985	437	0 %	790241.9

Table 8.26: Branch & Price – IMUCP instances – random demand

Conclusions

EXPERIMENTAL SUMMARY

Even if the UCP has been extensively studied from an experimental point of view, it remains hard to solve. This is due to the instance sizes, to the problem's compound structure featuring several hard combinatorial problems, and to symmetry issues. In this thesis, we choose to focus on the MUCP, which is the core structure of the real-world UCP and is already hard to solve by commercial solvers.

We propose several combinatorial techniques to solve difficult instances of the MUCP and its variants, namely the IMUCP, the MUCP with identical production units featuring ramp-constraints or not, and the MUCP with tight-production-range units. This experimental summary selects the most promising techniques for the resolution of the MUCP, depending on instance characteristics and on the features of the ILP solver used.

Intra-site constraints Intra-site constraints introduce a coupling between units located on the same production site. When added to the (x,u) formulation, they do not make MUCP instances harder to solve by Cplex. However, these constraints may interfere with classical decomposition schemes, where demand constraints are dualized and each subproblem correspond to a single thermal unit. It is shown in Chapter 8 that intra-site constraints have a very limited impact on the optimal value, and can be dualized alongside demand constraints without hindering the convergence of the decomposition scheme.

Symmetries Experimentations in Chapters 4, 6 and 7 show that symmetrical MUCP instances, with or without ramp-constraints, can barely be handled by Cplex. In the non-ramp-constrained case, the MUCP featuring identical units can be efficiently solved by aggregation of (x,u) variables. In the ramp-constrained case, the problem becomes much harder to solve. In this case, as aggregation is not possible anymore, symmetries and sub-symmetries can be handled with sub-symmetry-breaking inequalities, or with orbitopal fixing for the full sub-orbitope, up to sizes (n,T) = (30,96) and (60,48). If the problem is to be solved by default Cplex, *i.e.*, without any callback deactivating Cplex's features, the adjunction of sub-symmetry-breaking inequalities is recommended. Note that orbitopal fixing outperforms default Cplex on very symmetrical non-ramp-constrained instances such as $(n,T) = (60,48)$. This suggests that orbitopal fixing, even

though impeded by callbacks, could outperform sub-symmetry-breaking inequalities on very symmetrical ramp-constrained instances.

When the problem is solved by Callback Cplex, or with a solver not penalized (as much as Cplex) by callbacks, orbitopal fixing incurs almost no additional computational cost. In all these cases, it may prove more efficient to use orbitopal fixing instead of sub-symmetry-breaking inequalities.

Tight-production-range units Experimental results in Chapter 4 demonstrate that tight-production-range (TPR) instances are already very hard to solve, even in the non-symmetrical case. This is the case of TPR75 instances, such that $P^i_{min} = 75\%\, P^i_{max}$ for each unit i. These instances are quite similar in their structure to ramp constrained MUCP instances, where at each time t the possible power output is in a restricted interval. TPR100 instances, such that $P^i_{min} = P^i_{max}$ for each unit i, are also extremely difficult. These instances are close in their structure to MUCP instances featuring finite-power-output units. The main difficulty of TPR75 and TPR100 instances lies in the dynamic coupling of T knapsack problems, and can be alleviated using interval up-set inequalities.

It is shown in Chapter 4 that these inequalities used as cuts are particularly efficient on TPR75 instances. Therefore, a natural perspective would be to perform a polyhedral analysis of the ramp-constrained MUCP, in particular to lift interval up-set inequalities to the ramp-constrained case, and then to assess the experimental impact on ramp-constrained MUCP instances.

TPR100 instances remain very hard to solve, as $(n,T) = (10,96)$ or $(10,48)$ TPR100 instances cannot be handled by Cplex within the time limit of one hour. We observed in Chapter 8 that the time decomposition formulation with interval up-set inequalities provides the best lower bounds for TPR100 (I)MUCP instances. Improvement of the underlying column generation algorithm, as well as more dedicated branching rules, may be the ingredients of an efficient resolution algorithm for these particular instances. As the time decomposition formulation already includes every knapsack inequality, the integrality gap could be tightened by valid inequalities relying on the time coupling of T demand constraints. A TPR100-dedicated separation algorithm for interval up-set inequalities, as well as further analysis of the MUCP polytope, would therefore be useful to handle these difficult instances.

Solving the real-world UCP While the MUCP is already hard to solve, the real-world UCP is an even more challenging problem, due to its heterogeneous nature involving various combinatorial structures. Furthermore, many variants of the UCP feature non-linearities that impair the resolution process. It appears in the literature that the start-up cost is an exponential function of the unit's down time, and the production cost a quadratic function of the power produced. A question would be whether these non-linearities can be efficiently handled by non-linear solvers, and to what extent approximating the non-linearities impacts the optimal value. Dedicated techniques are likely to be needed.

As for combinatorial issues arising in the UCP, this thesis proposes several techniques to cope with symmetries, as well as to handle dynamically coupled knapsack constraints. In the perspective of solving large-scale real-world UCP instances, the latter techniques, used within a time decomposition structure, and in combination with appropriate techniques to handle non-linear aspects, could lay the bases of an efficient solver. One key challenge will be to cope with heterogeneous units featuring various technical constraints. As it is, the time-based decomposition could still provide a lower bound. The unit-subset-based decomposition could also be useful to handle heterogeneous units, as the corresponding column generation algorithm converges very fast. Valid inequalities would then be needed to improve the dual bound.

CONCLUSION AND PERSPECTIVES

Due to its practical relevance, the UCP has been constantly studied from both a research and practical point of view. In its core structure, the UCP reduces to the particular structure – we refer to as MUCP – induced by the coupling of demand and min-up/down constraints. Its complexity lies not so much in the knapsack embedded constraints as in the dynamics introduced by min-up/down constraints. As shown in Chapter 2, the UCP is a strongly (resp. weakly) NP-hard problem when relaxing the former (resp. latter) constraints. This result emphasizes that the combinatorial issues introduced by the min-up/down constraints should be specifically tackled when solving the UCP.

We propose a polyhedral study of the MUCP with n production units. We first compare various formulations for the MUCP, showing that the linear relaxation of any demand-coupling formulation is less than or equal to that of the classical formulation ($F^n_{x,u}$). In our subsequent study of the polytope associated to this formulation, we translate the classical extended cover inequalities of the knapsack polytope to obtain the up-set inequalities for the MUCP polytope. We generalize these up-set inequalities to obtain the interval up-set inequalities. This new class of valid inequalities is more dedicated to the MUCP as it captures the coupling between knapsack-like demand constraints and dynamic min-up/min-down constraints. We completely describe the cases in which these inequalities are valid, and we also characterize the facet defining cases in a restricted polytope. We devise an efficient Branch & Cut algorithm in which up-set and interval up-set inequalities are used as cuts.

Further theoretical questions about interval up-set inequalities would be to find their Chvátal-Gomory rank, and to characterize the cases in which they define facets of the dominant MUCP polytope. As pointed out in Section 3.4.3, multiple generalizations of interval up-set inequalities could lead to other facet defining inequalities. More generally, other classes of facets could be introduced for the n-unit MUCP polytope. An interesting question would be whether the inequalities defining these facets could be expressed more easily in a disaggregated variable space, as flow or interval variable space from formulations (F^n-*Flow*) and (F^n-*Int*). Another future work would be to study polyhedral aspects of the ramp-constrained MUCP. The ramp-constrained MUCP is close, in its structure, to the TPR75 instances of the MUCP on which interval up-set inequalities are particularly effective. Therefore, it may be useful to lift interval

up-set inequalities to the ramp-constrained case. As ramp constraints can be seen as big-M constraints, techniques to reformulate such constraints could also be applied to obtain a tight description of the ramp-constrained MUCP polytope.

Symmetries arise in the MUCP from the presence of identical unit. It is well known that symmetries in integer programs deeply impair their resolution by Branch & Bound.

We introduce a theoretical framework to simultaneously account for both the problem's symmetries and the symmetries arising in solutions subsets, defined as sub-symmetries. In particular, a condition to select a valid set of representatives in the presence of multiple sub-symmetry groups is given. We propose two flexible full symmetry-breaking techniques to handle symmetries and sub-symmetries in any integer program whose (sub-)symmetry groups are symmetric groups acting on (sub-)columns of the solution matrix.

The first proposed technique is a linear time orbitopal fixing algorithm for the full orbitope. This algorithm is proven to be optimal, in the sense that at any node in the B&B tree, the algorithm fixes all variables that can be fixed with respect to the lexicographical order.

This orbitopal fixing algorithm can be applied to both orbitope and sub-orbitope structures. Experimental results on MUCP instances show that this technique is competitive with commercial solvers like Cplex and state-of-the-art techniques like modified orbital branching (MOB).

This suggests as a perspective that a simultaneous orbitopal fixing algorithm for the full sub-orbitope could be found from the proposed sequential algorithm. Another perspective is to extend orbitopal fixing to full orbitopes under other group actions, for example the cyclic group. An extension to a full orbitope featuring integer entries can also be possible. An alternative approach to handle symmetries related to the symmetric or the cyclic group would be to find a new set of representatives whose convex hull would be easier to describe than the full orbitope.

The second technique we introduce corresponds to sub-symmetry-breaking inequalities, handling the symmetries arising from a collection of sub-symmetric solution subsets. These inequalities may require to introduce one additional variable per solution subset considered. The corresponding sub-symmetry-breaking inequalities are full symmetry-breaking. Experimental results for the ramp-constrained MUCP show that these sub-symmetry-breaking inequalities outperform all state-of-the-art symmetry-breaking formulations.

One perspective is to study how the framework presented could be automated, so that sub-symmetric subsets are automatically detected and additional variables automatically constructed. For the ramp-constrained MUCP, another perspective is to use the proposed framework to derive new sub-symmetry-breaking inequalities for ready-to-shut-down sub-symmetries.

There is a wide range of problems featuring all-column-permutation symmetries or sub-symmetries, on which it would be desirable to analyze the effectiveness of the symmetry-breaking techniques presented in this thesis. Among such problems are many variants of the UCP, covering problems, as well as some bin packing variants where one item can be placed in multiple bins. The question will be to determine which of the two techniques, orbitopal fixing or sub-symmetry-

breaking inequalities, will be the most efficient on each problem. Sub-symmetries-inequalities can easily be implemented in an ILP model, at the expense of adding some variables and some related inequalities to the formulation. While orbitopal fixing requires the use of a Callback, potentially hindering the performance of the ILP solver, its main advantage is that no additional variable nor inequality must be added to the formulation. Especially when the inequalities that must be added are difficult to manage, or even difficult to derive from the original variables, orbitopal fixing could come in handy and outperform sub-symmetry-breaking inequalities.

Classically, the UCP is solved via a unit-based decomposition where the demand constraint is dualized. A time-based decomposition could represent an alternative to this classical scheme. Indeed, from the study of various decomposition structures for the MUCP, it appears that the time decomposition, from the good quality bound it provides, is the most efficient structure within a Branch & Price framework.

Assessing the pertinence of time decomposition in the presence of heterogeneous units featuring specific technical constraints is left for further research. In any case, using a Branch & Price framework appears to be an attractive perspective. The separation of interval up-set inequalities proves to be useful in this context. In the presence of identical units, the time decomposition formulation naturally features symmetries which, as a perspective, could be handled using orbitopal fixing for the full (sub-)orbitope.

BIBLIOGRAPHY

[1] M. F. ANJOS, *Recent progress in modeling unit commitment problem*, in: L.F. Zuluaga, T. Terlaky (Eds.), Modeling and Optimization: Theory and Applications, Springer Proceedings in Mathematics & Statistics, 62 (2013), pp. 1–29.
1.2.7

[2] J. M. ARROYO AND A. J. CONEJO, *Optimal response of a thermal unit to an electricity spot market*, IEEE Transactions on Power Systems, 15 (2000), pp. 1098–1104.
1.2.7

[3] J. M. ARROYO AND A. J. CONEJO, *Modeling of start-up and shut-down power trajectories of thermal units*, IEEE Transactions on Power Systems, 19 (2004), pp. 1562–1568.
1.2.7

[4] E. BALAS, *Facets of the knapsack polytope*, Mathematical programming, 8 (1975).
3.4.1, 3.4.1, 4.1.1

[5] J. F. BARD, *Short-term scheduling of thermal-electric generators using lagrangian relaxation*, Operations Research, 36 (1988), pp. 756–766.
1.2.2, 2.3, 8.1

[6] C. BARNHART, E. JOHNSON, G. NEMHAUSER, M. SAVELSBERGH, AND P. VANCE, *Branch-and-price: Column generation for solving huge integer programs*, Operations research, 46 (1998), pp. 316–329.
5.2

[7] S. BAUM AND L. E. TROTTER, *Integer rounding and polyhedral decomposition for totally unimodular systems*, in Optimization and Operations Research, Springer, 1978, pp. 15–23.
1.3, 5.4

[8] P. BENDOTTI, P. FOUILHOUX, AND C. ROTTNER, *The min-up/min-down unit commitment polytope*, Journal of Combinatorial Optimization, 36 (2018), pp. 1024–1058.
3, 4

[9] ——, *On the complexity of the unit commitment problem*, Annals of Operations Research, (2018).
2

[10] ——, *Orbitopal fixing for the full (sub-)orbitope and application to the unit commitment problem*, Optimization Online, (2018).
http://www.optimization-online.org/DB_HTML/2017/10/6301.html.
6

[11] ——, *Sub-symmetry-breaking inequalities and application to the Unit Commitment Problem*, (2018).
https://hal.archives-ouvertes.fr/hal-01876358.
7

[12] A. BORGHETTI, A. FRANGIONI, F. LACALANDRA, AND C. A. NUCCI, *Lagrangian heuristics based on disaggregated bundle methods for hydrothermal unit commitment*, IEEE Transactions on Power Systems, 18 (2003), pp. 313–323.
8.1

[13] M. CARRION AND J. M. ARROYO, *A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem*, IEEE Transactions on Power Systems, 21(3) (2006), pp. 1371–1378.
1.2.4, 1.2.4, 1.2.4, 1.2.4, 1.2.7

[14] P. DAMCI-KURT, S. KÜÇÜKYAVUZ, D. RAJAN, AND A. ATAMTÜRK, *A polyhedral study of production ramping*, Mathematical Programming, 158 (2016), pp. 175–205.
1.2.6

[15] G. B. DANTZIG, *Maximization of a linear function of variables subject to linear inequalities*, Cowles comission, New York, (1951).
1.1

[16] J. DESROSIERS AND M. E. LÜBBECKE, *A primer in column generation*, in Column generation, Springer, 2005, pp. 1–32.
1.1.3

[17] ——, *Branch-price-and-cut algorithms*, Encyclopedia of Operations Research and Management Science. John Wiley & Sons, Chichester, (2011), pp. 109–131.
1.1.3

[18] L. DUBOST, R. GONZALEZ, AND C. LEMARÉCHAL, *A primal-proximal heuristic applied to the french unit-commitment problem*, Mathematical programming, 104 (2005), pp. 129–151.
1.2.1, 1.2.1, 1.2.2, 2.3, 8.1

[19] J. EDMONDS AND R. GILES, *A min-max relation for submodular functions on graphs*, in Annals of Discrete Mathematics, vol. 1, Elsevier, 1977, pp. 185–204.
1.1

[20] W. FAN, X. GUAN, AND Q. ZHAI, *A new method for unit commitment with ramping constraints*, Electric Power Systems Research, 62 (2002), pp. 215–224.
1.2.2, 2.3

[21] M. FISCHETTI, L. LIBERTI, D. SALVAGNIN, AND T. WALSH, *Orbital shrinking: Theory and applications*, Discrete Applied Mathematics, 222 (2017), pp. 109–123.
5.2.6

[22] A. FRANGIONI, *About Lagrangian methods in integer optimization*, Annals of Operations Research, 139 (2005), pp. 163–193.
1.2.2

[23] A. FRANGIONI AND C. GENTILE, *Perspective cuts for a class of convex 0–1 mixed integer programs*, Mathematical Programming, 106 (2006), pp. 225–236.
1.2.7

[24] A. FRANGIONI AND C. GENTILE, *Solving nonlinear single-unit commitment problems with ramping constraints*, Operations Research, 54 (2006), pp. 767–775.
1.2.2, 2.3

[25] A. FRANGIONI AND C. GENTILE, *New MIP formulations for the single-unit commitment problems with ramping constraints*, IASI Research Report 15-06, (2017).
1.2.6

[26] A. FRANGIONI, C. GENTILE, AND F. LACALANDRA, *Tighter approximated MILP formulations for unit commitment problems*, IEEE Transactions on Power Systems, 24 (2009), pp. 105–113.
1.2.7

[27] E. J. FRIEDMAN, *Fundamental domains for integer programs with symmetries*, in Combinatorial Optimization and Applications: First International Conference, COCOA 2007, Xi'an, China, August 14-16, 2007. Proceedings, A. Dress, Y. Xu, and B. Zhu, eds., Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 146–153.
5.2, 5.2.1

[28] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability*, W. H. Freeman and Company, New York, 1979.
1.1, 2.1

[29] I. GENT, T. KELSEY, S. LINTON, I. MCDONALD, I. MIGUEL, AND B. SMITH, *Conditional symmetry breaking*, Proc. 8th International Conference on Principles and Practice of Constraint Programming, (2005), pp. 256–270.
5.5

[30] I. P. GENT, T. KELSEY, S. A. LINTON, J. PEARSON, AND C. M. RONEY-DOUGAL, *Groupoids and conditional symmetry*, Proc. 9th International Conference on Principles and Practice of Constraint Programming, (2007), pp. 823–830.
5.5

[31] C. GENTILE, G. MORALES-ESPANA, AND A. RAMOS, *A tight MIP formulation of the unit commitment problem with start-up and shut-down constraints*, EURO Journal on Computational Optimization, (2016), pp. 1–25.
1.2.6

[32] A. M. GEOFFRION, *Lagrangian relaxation for integer programming*, in 50 Years of Integer Programming 1958-2008, Springer, 2010, pp. 243–281.
1.4

[33] A. GLEIXNER, L. EIFLER, T. GALLY, G. GAMRATH, P. GEMANDER, R. L. GOTTWALD, G. HENDEL, C. HOJNY, T. KOCH, M. MILTENBERGER, B. MÜLLER, M. E. PFETSCH, C. PUCHERT, D. REHFELDT, F. SCHLÖSSER, F. SERRANO, Y. SHINANO, J. M. VIERNICKEL, S. VIGERSKE, D. WENINGER, J. T. WITT, AND J. WITZIG, *The scip optimization suite 5.0*, Tech. Rep. 17-61, ZIB, Takustr.7, 14195 Berlin, 2017.
8.4.1

[34] M. GRÖTSCHEL, L. LOVÁSZ, AND A. SCHRIJVER, *Geometric algorithms and combinatorial optimization*, vol. 2, Springer Science & Business Media, 2012.
1.1.1, 1.2.5

[35] L. C. GROVE AND C. T. BENSON, *Finite reflection groups*, vol. 99, Springer Science & Business Media, 1996.
5.2.1

[36] Y. GUAN, K. PAN, AND K. ZHOU, *Polynomial time algorithms and extended formulations for unit commitment problems*, IISE Transactions, 50 (2018), pp. 735–751.
1.2.2

[37] A. J. HOFFMAN AND J. B. KRUSKAL, *Integral boundary points of convex polyehdra*, Princeton University Press, 1956, pp. 223–246.
1.2

[38] C. HOJNY AND M. E. PFETSCH, *Polytopes associated with symmetry handling*, Mathematical Programming, (2018), pp. 1–44.
5.2, 5.2.2, 5.3.3

[39] R. JANS, *Solving lot-sizing problems on parallel identical machines using symmetry-breaking constraints*, INFORMS Journal on Computing, 21 (2009), pp. 123–136.
5.3.1

[40] V. KAIBEL AND A. LOOS, *Branched polyhedral systems*, in Proceedings of the 14th International IPCO Conference on Integer Programming and Combinatorial Optimization, Springer-Verlag, 2010.
5.3.3

[41] V. KAIBEL, M. PEINHARDT, AND M. E. PFETSCH, *Orbitopal fixing*, Discrete Optimization, 8 (2011), pp. 595–610.
5.2, 5.2.5, 5.2.5, 5.3.3, 6

[42] V. KAIBEL AND M. E. PFETSCH, *Packing and partitioning orbitopes*, Mathematical Programming, 114 (2008), pp. 1–36.
5.2, 5.3.1, 5.3.3

[43] K. KAPARIS AND A. N. LETCHFORD, *Separation algorithms for 0-1 knapsack polytopes*, Mathematical Programming, 124 (2010), pp. 69–91.
4.1.1, 4.1.1

[44] N. KARMARKAR, *A new polynomial-time algorithm for linear programming*, in Proceedings of the sixteenth annual ACM symposium on Theory of computing, ACM, 1984, pp. 302–311.
1.1

[45] R. KARP AND C. PAPADIMITRIOU, *On linear characterizations of combinatorial optimization problems*, SIAM Journal on Computing, 11 (1982), pp. 620–632.
1.1.1

[46] L. G. KHACHIYAN, *Polynomial algorithms in linear programming*, USSR Computational Mathematics and Mathematical Physics, 20 (1980), pp. 53–72.
1.1

[47] K. KIM, A. BOTTERUD, AND F. QIU, *Temporal decomposition for improved unit commitment in power system production cost modeling*, IEEE Transactions on Power Systems, (2018).
8.1

[48] B. KNUEVEN AND J. OSTROWSKI, *A novel matching formulation for startup costs in unit commitment*, Optimization Online, (2017).

http://www.optimization-online.org/DB_FILE/2017/03/5897.pdf.
1.2.7

[49] B. KNUEVEN, J. OSTROWSKI, AND J. WANG, *Generating cuts from the ramping polytope for the unit commitment problem*, Optimization Online, (2016).
http://www.optimization-online.org/DB_HTML/2015/09/5099.html.
1.2.5, 1.2.6, 7.4.2

[50] B. KNUEVEN, J. OSTROWSKI, AND J. P. WATSON, *Exploiting identical generators in unit commitment*, IEEE Transactions on Power Systems, 33(4) (2017), pp. 4496–4507.
1.2.5, 5.2, 5.2.6, 5.4, 5.4, 7, 7.5.1, 8.7.1

[51] J. LEE, J. LEUNG, AND F. MARGOT, *Min-up/min-down polytopes*, Discrete Optimization, 1 (2004), pp. 77–85.
1.2.6, 1.2.6, 1.2.6

[52] C. LEMARÉCHAL, C. SAGASTIZÁBAL, F. PELLEGRINO, AND A. RENAUD, *Bundle methods applied to the unit-commitment problem*, in System modelling and optimization, Springer, 1996, pp. 395–402.
8.1

[53] L. LIBERTI, *Reformulations in mathematical programming: automatic symmetry detection and exploitation*, Mathematical Programming, 131 (2012), pp. 273–304.
5.2

[54] L. LIBERTI AND J. OSTROWSKI, *Stabilizer-based symmetry breaking constraints for mathematical programs*, Journal of Global Optimization, 60 (2014), pp. 183–194.
5.2

[55] R. M. LIMA AND A. Q. NOVAIS, *Symmetry breaking in MILP formulations for Unit Commitment problems*, Computers & Chemical Engineering, 85 (2016), pp. 162–176.
5.3.1, 5.4, 7, 7.5.1

[56] A. LOOS, *Describing Orbitopes by Linear Inequalities and Projection Based Tools*, PhD thesis, Universität Magdeburg, 2011.
5.3.3, 7.3, 7.3

[57] J. A. LÓPEZ, J. L. CECILIANO-MEZA, AND I. G. MOYA, *The challenges of the UCP for real-life small-scale power systems*, International Journal of Electrical Power & Energy Systems, 71 (2015), pp. 112–122.
1.2.7

[58] M. E. LÜBBECKE, *Column generation*, Wiley Encyclopedia of Operations Research and Management Science, John Wiley and Sons, Chichester, UK, (2010).
1.1.3

[59] P. MALKIN, *Minimum runtime and stoptime polyhedra*, Private Communication, (2003).
1.5, 1.2.5

[60] F. MARGOT, *Pruning by isomorphism in Branch-and-Cut*, in Proceedings of the 8th International IPCO Conference on Integer Programming and Combinatorial Optimization, London, UK, 2001, Springer-Verlag, pp. 304–317.
5.2.3

[61] F. MARGOT, *Exploiting orbits in symmetric ILP*, Mathematical Programming, 98 (2003), pp. 3–21.
5.2, 5.2.3

[62] ——, *Symmetry in integer linear programming*, in 50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art, M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, eds., Springer, Berlin, Heidelberg, 2010, pp. 647–686.
5.2.1, 5.2.1, 5.3.1, 5.3.1, 5.3.1, 6.2.2

[63] R. R. MEYER, *On the existence of optimal solutions to integer and mixed-integer programming problems*, Mathematical Programming, 7 (1974), pp. 223–235.
1.1.1

[64] M. MINOUX AND M. GONDRAN, *Graphes et algorithmes*, Lavoisier, 4th ed., 2009.
1.2.5

[65] G. MORALES-ESPANA, C. GENTILE, AND A. RAMOS, *Tight MIP formulations of the power-based unit commitment problem*, OR Spectrum, 37 (2015), pp. 929–950.
1.2.6, 1.2.7

[66] G. MORALES-ESPANA, J. M. LATORRE, AND A. RAMOS, *Tight and compact MILP formulation for the thermal unit commitment problem*, IEEE Transactions on Power Systems, 28(4) (2013), pp. 4897–4908.
1.2.6

[67] ——, *Tight and compact MILP formulation of start-up and shut-down ramping in unit commitment*, IEEE Transactions on Power Systems, 28(2) (2013), pp. 1288–1296.
1.2.7

[68] J. A. MUCKSTADT AND S. A. KOENIG, *An application of lagrangian relaxation to scheduling in power-generation systems*, Operations research, 25 (1977), pp. 387–403.
1.2.2, 8.1

[69] G. L. NEMHAUSER AND L. A. WOLSEY, *Integer programming and combinatorial optimization*, Wiley, Chichester. GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992). Constraint Classification for Mixed Integer Programming Formulations. COAL Bulletin, 20 (1988), pp. 8–12.
1.1

[70] J. OSTROWSKI, M. ANJOS, AND A. VANNELLI, *Modified orbital branching for structured symmetry with an application to unit commitment*, Mathematical Programming, 150 (2015), pp. 99–129.
5.3.2, 5.3.2, 5.4, 5.4, 5.5.3, 6, 6.4.3, 7.4.2, 7.5.1

[71] J. OSTROWSKI, M. F. ANJOS, AND A. VANNELLI, *Tight mixed integer linear programming formulations for the unit commitment problem*, IEEE Transactions on Power Systems, 27(1) (2012), pp. 39–46.
1.2.6, 1.2.7, 7.4.2

[72] J. OSTROWSKI, J. LINDEROTH, F. ROSSI, AND S. SMRIGLIO, *Constraint orbital branching*, in Integer Programming and Combinatorial Optimization: 13th International Conference, IPCO 2008 Bertinoro, Italy, May 26-28, 2008 Proceedings, A. Lodi, A. Panconesi, and G. Rinaldi, eds., Springer Berlin Heidelberg, 2008, pp. 225–239.
5.2.4

[73] ——, *Orbital branching*, Mathematical Programming, 126 (2011), pp. 147–178.
5.1, 5.2, 5.2.4

[74] N. P. PADHY, *Unit commitment-a bibliographical survey*, IEEE Transactions on Power Systems, 19 (2004), pp. 1196–1205.
1.2.2

[75] K. PAN AND Y. GUAN, *A polyhedral study of the integrated minimum-up/-down time and ramping polytope*, Optimization Online, (2016).
`http://www.optimization-online.org/DB_HTML/2015/08/5070.html`.
1.2.6, 7.4.2

[76] K. PAN, Y. GUAN, J. P. WATSON, AND J. WANG, *Strengthened MILP formulation for certain gas turbine unit commitment problems*, IEEE Transactions on Power Systems, 31 (2016), pp. 1440–1448.
1.2.6, 2.3

[77] C. PAPADIMITRIOU AND K. STEIGLITZ, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
2.5

[78] Y. POCHET AND L. A. WOLSEY, *Production planning by mixed integer programming*, Springer Science & Business Media, 2006.
1.2.5

[79] D. RAJAN AND S. TAKRITI, *Minimum up/down polytopes of the unit commitment problem with start-up costs*, IBM Research Report, (2005).
1.2.5, 1.2.5, 1.5, 1.2.6, 3.1, 3.2, 1, 8.2

[80] A. RENAUD, *Daily generation management at Electricité de France: From planning towards real time*, IEEE Transactions on Automatic Control, 38(7) (1993), pp. 1080–1093.
1.2.1, 1.2.2, 2.3

[81] RTE, *Mémento de la sûreté du système électrique*, (2004).
1.2.1

[82] G. SHEBLE AND G. FAHD, *Unit commitment literature synopsis*, IEEE Transactions on Power Systems, 9 (1994), pp. 128–135.
1.2.2

[83] C. C. SIMS, *Computational methods in the study of permutation groups*, in Computational Problems in Abstract Algebra, Pergamon, Oxford, 1970, pp. 169–183.
5.2.3

[84] M. TAHANAN, W. VAN ACKOOIJ, A. FRANGIONI, AND F. LACALANDRA, *Large-scale unit commitment under uncertainty*, 4OR, 13 (2015), pp. 115–171.
1.2.2

[85] S. TAKRITI, B. KRASENBRINK, AND L. S.-Y. WU, *Incorporating fuel constraints and electricity spot prices into the stochastic unit commitment problem*, Operations Research, 48 (2000), pp. 268–280.
1.2.6

[86] C. TSENG, *On Power System Generation Unit Commitment Problems*, PhD thesis, University of California, Berkeley, 1996.
2, 2.4

[87] J. M. VALÉRIO DE CARVALHO, *Using extra dual cuts to accelerate column generation*, INFORMS Journal on Computing, 17 (2005), pp. 175–182.
8.2.3

[88] W. VAN ACKOOIJ, I. DANTI LOPEZ, A. FRANGIONI, F. LACALANDRA, AND M. TAHANAN, *Large-scale unit commitment under uncertainty: an updated literature survey*, 2018.
1.2.2, 1.2.7

[89] F. VANDERBECK, *Branching in branch-and-price: a generic scheme*, Mathematical Programming, 130 (2011), pp. 249–294.
8.7.1

[90] F. VANDERBECK AND M. W. SAVELSBERGH, *A generic view of dantzig–wolfe decomposition in mixed integer programming*, Operations Research Letters, 34 (2006), pp. 296–306.
8.2.3

[91] F. VANDERBECK AND L. A. WOLSEY, *Reformulation and decomposition of integer programs*, in 50 Years of Integer Programming 1958-2008, Springer, 2010, pp. 431–502.
1.1.3

[92] W. YU, H. HOOGEVEEN, AND J. K. LENSTRA, *Minimizing makespan in a two-machine flow shop with delays and unit-time operations is NP-hard*, Journal of Scheduling, 7 (2004), pp. 333–348.
2.4.2