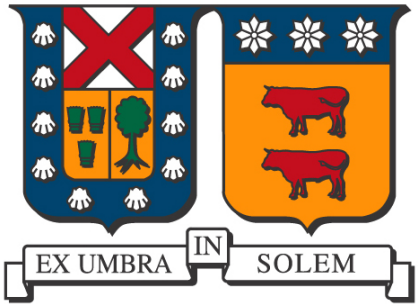


Universidad Técnica Federico Santa María

Department of Electrical Engineering

Cutting planes for the network flow formulation of the single unit commitment problem

Author:
José Ignacio Delgado Anguita

Thesis Director:
Dr. Alejandro Alberto Angulo Cárdenas

A thesis submitted in partial fulfillment of the requirements for the degree of

Magíster en Ciencias de la Ingeniería Eléctrica

March 3. 2026

CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): **X** Memoria o trabajo de título **X** Tesis de Postgrado

Título del trabajo: Cutting planes for the network flow formulation of the single unit commitment problem.

Nombre del candidato(a): José Ignacio Delgado Anguita.

Carrera / Grado: Ingeniería civil eléctrica/ Magister en ciencias de la ingeniería eléctrica.

Campus: Casa Central **Departamento:** Ingeniería eléctrica

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, **Alejandro Angulo Cardenas**, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

X El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por **(marcar una opción)**:

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

_____

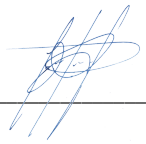
_____

_____

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 10/03/2026 **Firma:** _____



Estudiante o Candidato(a):

Fecha: 10/03/2026 **Firma:**

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Resumen

Este trabajo de tesis presenta una formulación alternativa basada en programación entera mixta para la operación de unidades generadoras basada en flujo en red, sujeta a restricciones de rampa. Si bien los límites de rampa son esenciales para garantizar una transición factible entre períodos consecutivos de tiempo, su inclusión afecta significativamente la relajación lineal de las formulaciones estándar, ampliando la región factible y deteriorando el desempeño computacional. Para abordar esta problemática, se desarrolló un estudio poliedral del espacio factible asociado a las restricciones de rampa, siguiendo una metodología basada en el método científico. Las observaciones iniciales se obtuvieron a partir de instancias generadas mediante el *software* PORTA, lo que permitió formular hipótesis sobre posibles desigualdades que pudieran definir facetas. Posteriormente, estas hipótesis fueron contrastadas mediante experimentos computacionales diseñados para determinar si las desigualdades candidatas definen efectivamente facetas del conjunto convexo asociado y para caracterizar con precisión las condiciones bajo las cuales dichas propiedades se satisfacen. A partir de este análisis, se propuso una formulación alternativa basada en flujo en red que incorpora las desigualdades que definen facetas derivadas del estudio. Su desempeño fue evaluado mediante experimentos computacionales en tres contextos diferentes: autodespacho de unidades generadoras, planificación de largo plazo del sistema eléctrico y auto despacho de plantas virtuales, los cuales se compararon con formulaciones existentes en la literatura que previamente han demostrado ser computacionalmente eficientes, en términos de tiempo de solución y calidad de la relajación lineal. Los resultados demuestran que la formulación propuesta logra relajaciones lineales más constreñidas, menor esfuerzo computacional y mejor escalabilidad tanto con respecto al horizonte de planificación como al número de unidades generadoras.

Abstract

This thesis work presents an enhanced mixed–integer linear programming (MILP) formulation for the network–flow (NF) based operation of generating units subject to inter–temporal ramping constraints. While ramping limits are essential to ensure feasible transitions between consecutive time periods, their inclusion significantly weakens the linear programming (LP) relaxation of standard formulations, enlarging the fractional feasible region and deteriorating computational performance. To address this issue, a polyhedral study of the ramping polytope was conducted following the scientific method. Initial observations were obtained from a large number of generated instances using the software PORTA, which enabled the formulation of hypotheses regarding potential facet–defining inequalities. These hypotheses were subsequently tested by verifying, through computational experiments, whether the candidate inequalities indeed define facets of the polyhedron, and the corresponding conditions. Building upon this analysis, an enhanced NF–based formulation incorporating the derived facet–defining inequalities was proposed. Its performance was evaluated through computational experiments across three different contexts: self-unit commitment (SUC), long–term power system planning, and virtual power plant (VPP) self–scheduling, and compared with state–of–the art formulations that have been previously shown to be computationally effective. The results demonstrate that the proposed formulation achieves tighter LP relaxations, lower computational effort and improved scalability with respect to both planning horizon and number of generating units.

Acknowledgements

A mi familia, toda mi gratitud y agradecimiento por su apoyo, su compañía y el cariño incondicional en cada momento, y en particular a mi padre y a mi madre por las enseñanzas y las oportunidades que me entragaron. A mis amigos por las risas, los buenos y malos momentos compartidos, y por estar siempre presentes. Y al profesor Alejando Angulo, por el apoyo y la orientanción necesaria para poder llevar a cabo este trabajo.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides a comprehensive summary of the thesis and situates the study within its research context. It introduces the unit commitment problem and discusses its inherent complexities, which motivate the methodological developments presented throughout the work. Next, it identifies existing gaps in the literature to clearly highlight the novelty and significance of this research. Finally, the specific research objectives are formally outlined, along with the methodology used to address them and the main contributions of this work to the state of the art.

1.1 Context and motivation

The unit commitment (UC) problem is a widely studied subject in the literature, where several solution approaches have been investigated, in terms of the formulations of the different elements of the problem, as well as the methods to solve the problem [3]. The UC problem is known to be NP–hard (Nondeterministic Polynomial-time hard), particularly due to the presence of binary variables in the commitment status combined with constraints that couple consecutive time periods. Leading to exponential growth in the feasible solution space as the number of units and time periods increases. The tractability of UC depends not only on the quality of the mathematical formulation but also on the problem–specific data, making both subjects significant in UC research.
In optimization, symmetry refers to the presence of multiple distinct variable assignments that represent equivalent solutions, yielding the same objective value and maintaining the feasibility [4]. In the UC context, symmetry occurs when elements of the formulation, such as generators, present similar or identical technical or economic parameters, leading to solutions that differ only by permutations of unit indices, thereby increasing the size of the branch–and–bound search tree without improving solution quality. To address this issue, symmetry–breaking techniques have been explored, where one of the first approaches, and the most naive, involves adding random perturbations to the parameters; however, this can sometimes negatively affect the performance of the algorithms. As well, in the context of UC, authors in [5] address the symmetry problem using symmetry–breaking constraints (SBC), which order the elements of the problem in a unique fashion, thereby reducing symmetry. Also, authors in [6] address the symmetry problem through an orbital–branching algorithm in the context of a thermal UC problem. Please refer to [7] for a more detailed overview of symmetry and symmetry–breaking techniques.
Furthermore, regarding the mathematical formulations, significant research has focused on developing strong linear relaxations for specific UC formulations. Notable progress has been made, especially regarding the minimum up and down polyhedron and the logical constraints associated with binary variables. However, the ramping constraints present challenges due to their intertemporal nature. This complexity affects the polyhedral structure of the formulation, resulting in larger integrality gaps and, consequently, longer solution times.

There exist in the literature convex-hull descriptions of the UC problem without ramping constraints, yielding ideal formulations whose linear relaxations are integral. In particular, the authors in [1] present the polyhedral analysis of the basic operation of the power–based UC problem. They demonstrate that the constraints related to generation limits, minimum up/down times, startup and shutdown trajectories, and startup and shutdown capabilities respectively gives facets to the convex hull of the formulation. Additionally, research has been conducted on the polyhedral structure of the ramping constraints, resulting in convex-hull descriptions for parts of the complete polyhedron. The authors in [8] derived the complete convex–hull description of the feasible region for the two–period case. For the general case, they defined valid inequalities to strengthen the formulation.
Given the considerations discussed above, it becomes evident that the UC problem is extremely challenging to solve, not only due to its large–scale, mixed-integer and strongly NP–hard [9] nature, but also because of the inherent underlying data. In particular, improving the mathematical formulation of the problem through stronger LP relaxations, remains a relevant line of research in this context toward obtaining better–quality solutions within reasonable computational effort.
Furthermore, advances in MILP solvers in recent year have made very important speed up in computational times [10, 11].
Consequently, given the aforementioned complexities of the unit UC problem, this thesis investigates the ramping polytope of the NF formulation for the generating units. The objective is to derive strong valid inequalities that tighten the feasible region of the LP relaxation, thereby enhancing the computational performance of state–of–the–art mixed–integer linear programming solvers, such as the widely used branch-and-bound (B&B) algorithm. In particular, a stronger LP relaxation reduces the root node gap, providing the B&B algorithm with a better starting point to exploit solver-specific tools, including heuristics and cutting planes.

1.2 Summary

This section provides an overview of the foundational hypotheses, methodological developments, and principal contributions from this thesis work.

1.2.1 Hypothesis of the work

The derivation and incorporation of problem-specific strong valid inequalities in the network flow formulation of the Single Unit Commitment problem will enhance the efficiency of solving the problem in terms of scalability of solution times. These inequalities will strengthen the root node of the branch-and-bound tree, reducing variable fractionality, minimizing branching steps, and ultimately decreasing the overall computational time required to achieve optimal integer solutions.

1.2.2 Problem statement and objectives

In the absence of ramping constraints, the standard NF–based formulation of the unit commitment problem admits a tight relaxation, in the sense that its LP relaxation is integral and therefore equivalent to the corresponding integer–programming (IP) formulation. With the addition of ramping constraints to the formulation, thereby coupling consecutive time periods, the structure of the underlying polyhedron is fundamentally altered, leading to the loss of integrality of the LP relaxation. As a result, the previously tight network–flow–based formulation no longer provides an exact description of the convex–hull of feasible unit commitment solutions, and fractional extreme points may arise.
In this sense, this work presents a comprehensive study of the ramping polytope for network–flow based formulations through the respective polyhedral analysis, aiming to obtain strong valid inequalities for the network–flow based formulation of the self–UC. The specific objectives are as follows:

1. Derive strong–valid inequalities in order to obtain a strong LP relaxation of the problem, through the polyhedral study of the ramping polytope.

2. After deriving the set of valid inequalities, a thorough examination of the nature of the inequality must be conducted in order to determine whether the inequality is only valid for the ramping polytope or if it also defines facets of the feasible region of the polyhedron being studied. Additionally, we need to identify the conditions under which the corresponding valid inequality is considered a facet.

3. After examining each valid inequality and defining the conditions under which it describes facets, we conduct computational experiments to evaluate the performance of the proposed constraints. These experiments focus in two relevant problems in the power system operation research literature, such as the UC problem, and the self–UC problem, evaluated both in different contexts.

1.2.3 Methodology

To achieve the objectives outlined in the previous section, the following methodology is adopted:

1. Build a self–UC problem based on the network–formulation of the generating units, following the approach proposed in [12]. The formulation is written in the mathematical programming language AMPL [13], because of the flexibility and solver–agnostic design, which enables the use of a wide range of state–of–the–art optimization solvers, including `Gurobi` [14], `CPLEX` [15, 16], `Knitro` [17], `Xpress` [18, 19], `Mosek` [20] among others.

2. Design an algorithm capable of determining the extreme points of a given polyhedron and, subsequently, deriving its hyperplane representation using the polyhedral analysis tool `PORTA` [21].

3. Conduct computational experiment comparing the enhanced formulation with the standard NF self–UC, and the classical 3*bin* formulation proposed in [22], in three different context relevant in the power system operation literature, such as the self–commitment of generating units problem in the day–ahead market, long–term power system planning, and finally the self–scheduling problem of a virtual power plant also in the day–ahead market. To ensure that the computational experiment is scenario–agnostic, we will consider various instance configurations, focusing on the parameters, sizes of the generating units, and the temporal horizon.

1.2.4 Contributions

The main contributions of the present thesis can be summarized as follows:

1. This work presents a polyhedral analysis of a particular polyhedron applied in the context of the network–flow formulation of generating units, particularly their ramping operations. It should be noted that the polyhedral analysis conducted in this thesis is not restricted to the specific application considered herein; rather, it can be extended to a wide range of domains involving networks subject to gradient constraints, such as transportation and traffic networks, supply chain and logistics, communication networks, among other fields.

2. This thesis proposes a strengthened LP relaxation for the NF–based UC formulation, achieved through the introduction of a novel family of valid inequalities. These inequalities are shown, under certain conditions, to define facets of the ramping polyhedron, thereby significantly enhancing the tightness of the relaxation and improving the overall computational tractability of the problem.

3. The proposed approach provides a systematic framework for deriving strong polyhedral results in a wide range of network–based optimization problems. By examining the structural properties of the underlying polyhedron, this framework facilitates the identification of facet–defining or valid inequalities by characterizing extreme points.

1.2.5 Document structure

The remainder of this document is organized as follows. Chapter 2 includes a background of the relevant theoretical foundations addressed in this thesis. It includes an extensive description of optimization in networks, as well as a review of different unit commitment formulations, with particular emphasis on those pertinent to this work, namely the NF–based formulation and the widely studied *3bin* model, alongside with thesis respective mathematical formulation. The chapter also presents the key concepts from mathematical optimization and set theory that are considered fundamental for understanding the polyhedral analysis conducted in this work, along with illustrative examples to facilitate their understanding. Subsequently, Chapter 3 presents the mathematical formulation of the proposed NF–based model, including a motivational example to illustrate the significance of the work and to justify its necessity. The chapter also includes a presentation of the polyhedron under study, as well as a detailed description of the methodology employed to perform the polyhedral analysis and derive facet-defining inequalities. Subsequently, the relevant polyhedral results are presented, including the trivial proofs, along with the mathematical formulation of the facet-defining inequalities and the corresponding proofs that establish the conditions under which each inequality defines a facet of the polyhedron. Chapter 3 presents the computational experiments that were conducted, along with the corresponding results and a detailed discussion and analysis of these findings. Finally, Chapter 5 presents the main conclusions of this thesis, along with a final discussion of the results, and potential directions for future research and improvements.

Chapter 2

Background

In the present chapter, the main topics related to the thesis work are reviewed. To better understand the relevance of the self unit commitment problem, a brief description of the structure of electricity markets is given. Then, relevant differences regarding the Unit Commitment problem are presented to provide a more comprehensive characterization of the problem presented. Different optimization models for the Unit Commitment are presented, including the network flow formulation, which is studied in this work, with a particular focus on mixed–integer linear formulations. Finally, an overview of polyhedral theory is presented, introducing existing cutting plane techniques and concluding with an overall review of the existent applications of polyhedral analysis in the context of the unit commitment problem.

2.1 Network optimization

In the present work, we propose formulating the operation of a thermal unit as a network with additional constraints, referred to in the literature as side constraints, i.e. additional linear inequalities that are not directly related to flow conservation, but must be satisfied by the solution. In particular, a shortest path problem with gradient constraints in each node of the network. This chapter summarizes some properties of network problems and presents the structure of the problem studied.

2.1.1 Graphs and flows

In this section, we introduce some of the basic definitions relating to graphs, paths, flows, and other relevant notions [23]. A directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ is formed by a finite set of nodes $\mathcal{N}$ and a set of arcs $\mathcal{A}$, where each arc is an ordered pair (i, j) of distinct nodes. The ordering implies that (i, j) and (j, i) represent different arcs. We say that an arc (i, j) originates at node i (source) and terminates at node j (sink). The degree of a node i is the number of arcs that are incident to it, regardless of direction. Graphs may be directed or undirected, in a directed graph, the order of the nodes in an arc is significant, while in an undirected graph, the arc simply connects two nodes without a specified direction.
A path in a directed graph is an ordered sequence of nodes $(n_1, n_2, ..., n_k)$ with $k \geq 2$ such that consecutive nodes are linked by arcs. Also, we denote by N_i^+ the set of a arcs entering the node i and N_i^- the set of a arcs leaving the node i. A flow assigns a real value x_{ij} to each arc $(i, j) \in \mathcal{A}$, representing the amount of some quantity that moves along that arc.
The divergence at a node i is defined as the total inflow minus the total outflow at that node. The vector of divergences over all nodes is called the divergence vector y, which can be expressed as an

N-dimensional vector derived from the flow vector.

$$y_i = \sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ij} - \sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ji}, \qquad \forall i \in \mathcal{A}. \tag{2.1}$$

A node i is called *source* and *sink* for the flow vector x if respectively, $y_i > 0$ and $y_i < 0$.

2.1.2 Network–flow models

Network flow models are a class of optimization problems that arise in numerous contexts such as supply chain logistics, transportation, telecommunications, and energy transmission. Because of their special structure, many network flow problems can be solved efficiently with specialized algorithms running in polynomial time [24, 25]. A network–flow optimization problem with linear cost function can be written as follows:

$$z = \min\left\{\boldsymbol{x} \mid \boldsymbol{c}^T\boldsymbol{x}, \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b},\ \boldsymbol{0} \leq \boldsymbol{x} \leq \boldsymbol{u},\ \boldsymbol{x} \in \mathbb{Z}^n\right\}, \tag{2.2}$$

where $\boldsymbol{c}$ is the cost vector, $\boldsymbol{A}$ is the coefficient matrix that includes the flux and divergence constraints, $\boldsymbol{b}$ is the vector containing the supply at node i, and $\boldsymbol{u}$ represents the upper bound on the flux variable $\boldsymbol{x}$. Note that the matrix of coefficients $\boldsymbol{A}$ is *totally unimodular* (TU), which implies that it always define polytopes with integer vertices. Since the formulation captures the convex hull of the feasible space, it can be easily solved using available commercial solvers.
Network–flow based formulations can be applied in a large variety of contexts [26]. In particular, one of the most discussed problem in this context is the *minimum cost flow problem* which can be broadly classified into four distinct groups [27, 28]: *i)* the shortest path problem, *ii)* the transportation problem, *iii)* the max–flow problem and *iv)* the assignment problem. Given a network with N nodes and A arcs, each of the problems involve:

i) **Shortest path problem:** the shortest path problem involves determining the shortest route between a source node (s) and a sink node (t) [29]. The goal is to minimize the total cost of the route, where each arc (i, j) has a designated cost denoted by a_{ij}. We may formulate the shortest path problem as following:

$$z_{sp} = \min\left\{\boldsymbol{x} \mid \sum_{(i,j)\in\mathcal{A}} a_{i,j}x_{i,j}, \sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ij} - \sum_{\{j|(j,i)\in\mathcal{A}\}} x_{ij} = \delta_i,\ \ x_{i,j} \geq 0\ \ \forall\,(i,j)\ \in\ \mathcal{A}\right\}, \tag{2.3}$$

where $\delta_i = \{1 \text{ if } i = s,\ -1 \text{ if } i = t,\ 0 \text{ otherwise}\}$, also, let x_{ij} be equal to 1 if (i, j) belongs to the optimal path P and 0 otherwise. Please note that none integer constraint on a variable may be imposed given that the matrix of coefficients is TU; as such, the optimal solution is always integer.

ii) **Transportation problem:** In such a problem, there are a set of nodes called sources (M), and a set of nodes called destinations (N). All arcs go from a source (i) to a destination (j). There is a per–unit cost (a_{ij}) on each arc. Each source has a supply of material (α_i), and each destination has a demand (β_j). We may then formulate the transportation problem as the linear program below.

$$\begin{aligned} z_{tp} = \min\Bigg\{\boldsymbol{x} \mid \sum_{(i,j)\in\mathcal{A}} a_{i,j}x_{i,j}, \sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ij} = \alpha_i\ \ \forall i \in M, \sum_{\{i|(i,j)\in\mathcal{A}\}} x_{ij} = \beta_j\ \ \forall j \in N, \\ \sum_{i\in M}\alpha_i = \sum_{j\in N}\beta_j, \quad 0 \leq \boldsymbol{x} \leq \min\{\alpha_i, \beta_j\}\ \ \forall\ (i,j) \in \mathcal{A}\Bigg\} \end{aligned} \tag{2.4}$$

The third constraint ensures that the quantity sent matches the quantity received. Additionally, the final constraint ensures that the goods sent do not exceed the production capacity of the source node or the demand of the respective destinations.

iii) **Assignment problem:** A special case of the transportation problem is the assignment problem which occurs when each supply is 1 and each demand is 1. Suppose there are n persons and n tasks that we have to match on a one–to–one basis, a benefit a_{ij} for matching person i with task j, and we want to maximize the total benefit, which corresponds to the optimal match between persons and tasks, subject to assignment constraints. We may then formulate the assignment problem as the linear program below.

$$z_{ap} = \min \left\{ \boldsymbol{x} \mid \sum_{(i,j)\in\mathcal{A}} a_{i,j}x_{i,j}, \sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ij} = 1 \ \forall i = 1, ..., n, \right. \\ \left. \sum_{\{i|(i,j)\in\mathcal{A}\}} x_{ij} = 1 \ \forall j = 1, ..., n, \ \boldsymbol{x} \geq 0 \right\} \tag{2.5}$$

The first constraint involves that the divergence of a person (node) i should be equal to 1, while the second constraint indicates that the divergence of a task (node) j should be equal to -1. Which implies that every person will be assigned one destination task and every task will have one destination.

iv) **Max–flow problem:** The max–flow problem objective is to move as much flow from a source node (s) to a sink node (t), subject to to capacity constraints on each arc. The given problem arises in several contexts such as highway system or communication networks. We can mathematically formulate the problem as following.

$$z_{tp} = \max \left\{ \boldsymbol{x} \mid \sum_{\{i|(i,t)\in\mathcal{A}\}} x_{i,j}, \right. \\ \sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ij} - \sum_{\{j|(j,i)\in\mathcal{A}\}} x_{ij} = 0 \ \forall \ i \in \mathcal{N} \text{ with } i \neq s \text{ and } i \neq t, \\ \left. 0 \leq x_{i,j} \leq c_{i,j} \ \ \forall \ (i,j) \in \mathcal{A} \right\} \tag{2.6}$$

Notice that the objective is to maximize the sum of arcs entering the sink node, but is equivalent to maximizing the sum of arcs leaving the source node.

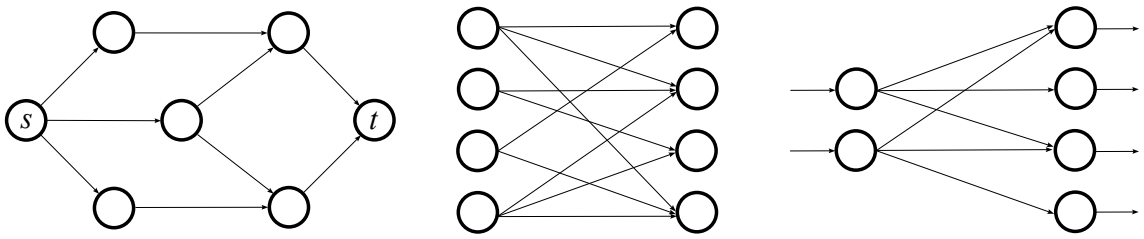


Figure 2.1: The Minimum Cost Flow Problem graphs, from left to right: a minimum cost flow and shortest path example network, a simple assignment network and a graph representation of a transportation problem.

2.2 UC models

The SUC problem involves determining the optimal self-dispatch schedule for a group of generating units over a specified planning horizon. This includes deciding when each unit should be turned *on* or *off*, as well as how much power it should generate while it is *on*, based on a given revenue

price, in order to maximize total income. Nevertheless, the unit commitment problem generalizes SUC to a system with large number of generating units. The goal is to determine the optimal combination of *on/off* states and production levels for all units in the system, such that total demand and reserve requirements are satisfied, network/operational constraints are respected, and total system cost is minimized.
The UC problem is a large-scale mixed-integer optimization problem that coordinates the operation of an entire power system, rather than a single or a reduced number of generators.

2.2.1 UC formulations

The formulation of a given MILP problem can be done in various ways, and improving the MILP formulation is a well-established research area. In this context, Rajan and Takriti [30], in 2005, investigated specific operational characteristics of generators to derive strong valid inequalities. Their focus was on the minimum up–and down–time constraints, from which they derived the strongest possible inequalities corresponding to facets of the convex hull of feasible operating schedules under the *3-bin* formulation. Consequently, their approach yields the tightest linear programming relaxation with respect to these constraints. Moreover, Carrion and Arroyo [31] proposed, in 2006, a formulation that reduces the number of binary variables to a single commitment variable representing the *on/off* status of each generator. This approach decreases the number of rows and columns in the formulation, thereby yielding a more compact model. However, it also affects the tightness of the resulting relaxation, as later analyzed by the same authors. Also in 2006, Frangioni and Gentile [32] proposed a dynamic programming algorithm to solve the single UC problem including ramping constraints, using general convex cost functions instead of quadratic functions approximated by piecewise methods that increase the solution times. The algorithm proposed by the authors builds upon an efficient approach for solving the SUC economic dispatch problem with ramping constraints. It has been shown to outperform mixed-integer quadratic programming (MIQP) solvers in solving SUC instances. Furthermore, the work presented by Frangioni and Gentile [33], in 2009, approximates the non–linear objective function that represents the total power production by means of valid inequalities, known as *perspective cuts*, rather than using piecewise linearizations. This approach yields a tighter linear relaxation of the objective function, thereby strengthening the formulation and improving solver efficiency. The work of Ostrowski et al. [2], in 2012 proposes a new family of ramping inequalities for one–and–two periods for the *3–bin* formulation, leading to a tighter characterization of the feasible region associated with generator operating schedules. In addition, the authors provide proofs regarding the polyhedral properties of these inequalities. Likewise, in 2013, Morales et al. [22] proposed what is now considered the state–of–the–art *3-bin* formulation for the unit commitment problem. This formulation is shown to be both tight and compact, as it reduces the search space and accelerates the solution process, thereby significantly lowering the computational burden. In 2015, Morales et al. [1] proposed a convex-hull formulation for the UC problem that captures the basic operational constraints of generators, including power limits, minimum up and down times as introduced in [30], and logical relation among the binary variables. The authors demonstrated that the formulation exactly matches the LP relaxation. Computational tests were conducted for both self-contained UC instances and network-constrained UC problems. Moreover, Kneuven et al. [34], in 2018, presented a perfect formulation for the single–UC problem applicable to a very general class of generating units, building upon the dynamic programming approach introduced by Frangioni and Gentile [32] in 2006. The authors employed the extended formulation to generate cuts aimed at strengthening the LP relaxation. The cuts were particularly effective in reducing the integrality gap of the formulation, especially for cases with a high impact of uncertainty. For a comprehensive and detailed review of UC formulations, the reader is referred to the work of Knueven et al. [35], which surveys the UC literature up to 2020.
Furthermore, the different methods of solving MILP problems has been extensively researched in the literature, leading to the emergence of various solution methods over the years as remarked by Padhy et al. in [36]. Among all the existent methodologies, we can find a technique called

exhaustive enumeration addressed in Hara et al. [37] and Lee et al. in [38], based on enumerating all possible combinations of the generating units and then the combinations that yield the least cost of operation are chosen as the optimal solution. Priority listing is another solution methodology for solving combinatorial mixed integer programs as discussed in Burns et al. [39] and Lee et al. in [40]. In the context of the UC problem, this method involves arranging the generating units according to an ascending operational cost curve and then using this predetermined order to ensure that system demand is met by intersecting the resulting supply and demand curves. As well, dynamic programming is another widely method for solving the UC problem as studied by Synder et al. in [41], and Pang et al. in [42]. In general terms, the approach aims to minimize the total running cost of supplying a total load of x megawatts (`MW`) of load using N generating units. This is achieved by determining the load y (in `MW`) assigned to a given unit n such that the remaining load $(x - y)$ `MW` is optimally allocated to the remaining $(N - 1)$ units, thereby ensuring that the total cost is minimized. One well known and extensively used method for solving MILP problems in optimization is the Branch and Bound algorithm. Authors in [43, 44] presented an approach for solving the UC problem using this technique. The method involves iteratively relaxing the master problem and solving various LP problems with additional constraints imposed on the branched variables until some criterion about time, explored nodes, or integrality gap, among other criteria, is met. The reader is referred to [45] for a more detailed review about UC solution approaches based in optimization techniques.

Furthermore, in the literature there exists also machine–learning (ML) techniques to solve the UC problems, including unsupervised learning, supervised learning and reinforcement learning approaches. In this context, Lin et al. in [46] proposed a ML-based approach to derive approximations of the MIP formulations that are close to the LP relaxation of the problem without directly solving the original problem. The authors achieve their goal through classification and regression tree models, as well as random forest models. Moreover, Xavier et al. in [47] proposed a ML–based approach in order to prove warm–start (i.e. good initial feasible solution) for MIP solvers based on information from previously solved instances. This approach also helps eliminate redundant constraints, resulting in more compact formulations and improved solution times.

The reader is referred to [48, 49] for a more detailed review of machine-learning–based approaches for solving unit commitment problems. Typically, ML–based methods are integrated with optimization techniques, often enhancing their performance by generating warm–start solutions that guide the solver and reduce computational effort. In addition, ML can help identify and eliminate redundant constraints based on patterns observed in previous solutions. Furthermore, ML can predict solver hyperparameters, such as branching rules, cut selection, or node processing order–based on instance features, among other potential improvements, while leaving the final decision-making process to the underlying optimization algorithm [50].

In particular, our study focuses on the network-flow–based formulation and the *3-bin* model, both of which are described in detail in the following sections. While the *3-bin* model has been extensively studied in the literature, the network-flow–based formulation has received comparatively less attention.

2.2.2 Network flow UC model

For the Network Flow Unit Commitment (NFUC), the network illustrates all possible scheduling operations for the units as presented in [12]. In this context, the nodes represent the number of periods each unit has been *off* or *on* during specific time periods, while the arcs indicate feasible transitions between consecutive time periods. Generally, the size of the network is determined horizontally by the time horizon of the problem and vertically by up and down times of the unit. The Figure 2.2 shows an example of the network representation for a single generator where the red path represents one of many feasible schedules. Note that the minimum up and down time of the unit are embedded in the network forcing them to be met.

At the extreme nodes of each period of the time horizon (i.e. $t = 0$) the node number indicates that the unit has been on or off for at least that number of periods. In the proposed example, the

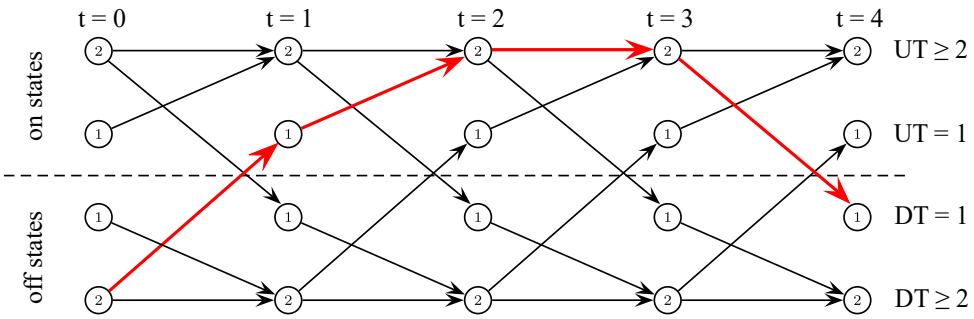


Figure 2.2: Single generator network with equal minimum up and down time, both set to a value of 2 periods, for planning horizon of four time periods.

unit has been offline for at least two periods at the start of the planning horizon. It is turned on at the first period, stays online for periods two and three, and shuts down in the last period.
Given a network that represents a generator $g \in \mathcal{G}$, let $g \in \mathcal{V}_g$ and $g \in \mathcal{E}_g$ be the set of nodes and arcs, respectively. Let $\mathcal{E}_g(t) \subseteq \mathcal{E}_g$ be the subset of arcs that arrives at a node or state at time $t \in \mathcal{T}$. Let x_e be a binary variable associated with the use of the arc $e \in \mathcal{E}_g$ and let z_e be a nonnegative continuous variable related to the power generated in the arc $e \in \mathcal{E}_g$. Let $\underline{P}_e$ and $\overline{P}_e$ be the minimum and maximum power parameters of the arc $e \in \mathcal{E}_g$. To ensure clarity in the mathematical notation, we will omit the subindex g in the following equations. The subsequent conditions must be satisfied:

$$\underline{P}_e x_e \leq z_e \leq \overline{P}_e x_e \qquad \forall e \in \mathcal{E} \tag{2.7}$$

Equation (2.7) limits the power of the arc if the arc is being used, throw mathematical operations on the presented equations, we can derive the variable $y_e \forall e \in \mathcal{E}$, wish limits are as shown below:

$$0 \leq y_e \leq x_e \qquad \forall e \in \mathcal{E} \tag{2.8}$$

Variable y_e represents the fraction of power produced above the minimum and is bounded by the binary variable x_e. Therefore, the power generation of a specific unit for each time period is expressed as follows:

$$\sum_{e \in \mathcal{E}(t)} (q_e x_e + s_e y_e) = p_{gt} \qquad \forall\, t \in \mathcal{T}, \tag{2.9}$$

where $q_e = \underline{P}_e$ and $s_e = \overline{P}_e - \underline{P}_e$ are preprocessed parameters of the network to simplify the notation. On the other hand, fixed costs, such as starting costs, can be included in the fixed costs of the network, while generation costs can be considered in the variable costs. Let CF_e and CV_e be the fixed and variable costs for the arc $e \in \mathcal{E}$. Then, the network cost is defined by the following function:

$$\sum_{e \in \mathcal{E}} (c_e x_e + f_e y_e) \tag{2.10}$$

As well, $c_e = q_e CV_e + CF_e$ and $f_e = s_e CV_e$ are preprocessed network parameters to simplify notation. The state of the unit at the start of the planning horizon is known, so the route is defined by flow or divergence constraints. Let $\delta^+(i)$ and $\delta^-(i)$ be the sets of arcs leaving and entering the node $i \in \mathcal{V}$. Let b_i be a parameter associated with node $i \in \mathcal{V}$, then the flow constraint at each node is as follows:

$$\sum_{e \in \delta^+(i)} x_e - \sum_{e \in \delta^-(i)} x_e = b_i \qquad \forall i \in \mathcal{V}_g \tag{2.11}$$

In particular, the parameter b_i only takes value 1 in the initial unit state in period $t = 0$. Since the parameter b_i has no value in all other states, the divergence constraint ensures that a single path is generated in the network.

To effectively model the operation of a generation unit, the model must include the ramping constraints. These constraints reflect the unit's ability to adjust its output between consecutive periods. Ramping limits ensure that the generation schedule adheres to the physical capabilities of the turbine, preventing changes in output that exceed the unit's technical ramp–up and ramp–down rates.

$$\sum_{e \in N_i^-} (\underline{P} x_e + \Delta_g y_e) - \sum_{e \in N_i^+} (\underline{P} x_e + \Delta_g y_e) \leq RU, \qquad \forall e \in \mathcal{E}, \tag{2.12a}$$

$$\sum_{e \in N_i^+} (\underline{P} x_e + \Delta_g y_e) - \sum_{e \in N_i^-} (\underline{P} x_e + \Delta_g y_e) \leq RD, \qquad \forall e \in \mathcal{E}. \tag{2.12b}$$

Note that prior to the inclusion of ramping capabilities, the constraints (2.8) to (2.11), along with the binary condition on the variables x_e, form a polyhedron whose coefficient matrix is totally unimodular (TU) [51]. Which means that the solution of the linear–programming (LP) problem exactly matches the integer–programming problem (IP), making the problem very easy to solve with standard techniques.
Nevertheless, adding ramping constraints (2.12) to the formulation changes the nature of the matrix A, resulting in a constraint matrix whose structure no longer preserves total unimodularity and thereby alters the polyhedral properties of the feasible region, resulting in a problem much harder to solve.
Consider now the complete formulation of the model for the NF–base SUC:

$$\textbf{NFSUC:} \quad \min_{x,y} \quad \sum_{g \in \mathcal{G}} \sum_{e \in \mathcal{E}_g} \left[\pi_e \left(q_e x_e + s_e y_e \right) - \mathcal{C}_e \left(x_e, y_e \right) \right] \tag{2.13a}$$

$$\text{s.t.} \quad (2.8), (2.11), (2.12) \tag{2.13b}$$

$$x_e \in \{0, 1\} \qquad \forall e \in \mathcal{E}, \tag{2.13c}$$

where π_e correspond to the energy price for the given set of edges, related to a specific time period. The cost $\mathcal{C}_e \left(x_e, y_e \right)$ includes the expenses incurred when changing from *off* to *on* status. This component, associated with the start–up operations of the unit, can be computed as follows:

$$\mathcal{C}^{SU} = \sum_{e \in \mathcal{E}^{\text{su}}} SU x_e. \tag{2.14}$$

In this equation, the set $\mathcal{E}^{\text{su}}$ includes all the arcs in the network that pertain to the start-up paths of the unit, specifically the arcs connecting a node labeled as *off* to a node labeled as *on*.
Furthermore, the UC problem is closely related to the SUC problem, differing primarily in their objective functions and operational context. Formally, the UC problem aims to minimize the total system operating cost, subject to the technical and operational constraints of the generating units, while ensuring that the total production satisfies system demand at each time period of the planning horizon. In the NF–based UC formulation, the demand satisfaction constraints takes the following form:

$$\sum_{g \in \mathcal{G}} \sum_{e \in \mathcal{E}_g(t)} (q_e x_e + s_e y_e) = D_t \quad \forall t \in \mathcal{T}, \tag{2.15}$$

where D_t correspond to the demand of the system in time period t. It is worth noting that this constraint can be extended to include non-served energy, allowing the formulation to account for situations in which demand is not fully met and providing a mechanism to penalize or limit load curtailment.

2.2.3 3-bin UC model

In this subsection, we present a detailed description of the *3-bin* model proposed in [22], which will serve as a benchmark for comparison with the network flow formulation of the UC and the

enhanced formulation of the former, proposed in this thesis work. Other MILP formulation of the UC problem exists, based primarily on complete enumeration of the feasible region [38, 37, 52]. However, since complete enumeration is not an efficient technique, these approaches were only able to solve relatively small instances.
The *3-bin* formulation, first proposed in [53] represents the state of a generator using three binary variables. The first variable, u_t, indicates the generator's status: it takes the value *1* if the generator is online and *0* if it is offline. The second variable, v_t, represents the start-up status of the unit, with a value of *1* if it starts at time t and *0* otherwise. Lastly, the variable w_t pertains to the shutdown operations of the generator, following the same logic as the binary variable for the start-up status. The logical constraint which relate these three binary variables is:

$$u_t - u_{t-1} = v_t - w_t \qquad \forall t \in \mathcal{T} \tag{2.16}$$

In addition, note that if the variable *u(t)* is defined as binary, equation (2.16) enforces variables *v(t)* and *w(t)* to take binary values even if defined as continuous.
To guarantee that the minimum–up (UT) and down times (DT) of the generator are satisfied, the authors in [30, 54], proposed the following turn–on and turn–off inequalities:

$$\sum_{i=t-UT+1}^{t} v_i \quad \leq \quad u_t \qquad t \in \{UT, ..., T\} \tag{2.17a}$$

$$\sum_{i=t-DT+1}^{t} w_i \quad \leq \quad 1 - u_t \qquad t \in \{DT, ..., T\} \tag{2.17b}$$

Equations (2.17a) and 2.17b represent the minimum up-time and down-time constraints of the generator. This means that the unit must remain on or off for at least UT and DT time periods before it can be shut down or started up, respectively. As shown by Malkin in 2003 [54] and Rajan and Takriti in 2005 [30], the constraints in (2.17) are facets of the convex hull of the up-time and down-time polytopes, which means that they provide the strongest bounds to formulate the minimum up and down times of a generator. For a detailed proof of this results, please refer to [30, 54].
Furthermore, for unit which minimum up time is equal to one period, the generation limit above the minimum power ($\underline{P}$) output considering the spinning reserve (r_t) are presented as follows:

$$p_t + r_t \leq (\overline{P} - \underline{P})u_t - \left(\overline{P} - SU\right) v_t \qquad \forall t \in \mathcal{T} \tag{2.18a}$$

$$p_t + r_t \leq (\overline{P} - \underline{P})u_t - \left(\overline{P} - SD\right) w_{t+1} \qquad \forall t \in \mathcal{T}, \tag{2.18b}$$

where the variable p_t is defined as the power output of the unit above the minimum. If the minimum up–time of the unit is greater than one period, the equations in (2.18) can be substituted with a more compact and tighter constraint that encompasses both.

$$p_t + r_t \leq (\overline{P} - \underline{P})u_t - \left(\overline{P} - SU\right) v_t - \left(\overline{P} - SD\right) w_{t+1}, \qquad \forall t \in \mathcal{T} \tag{2.19}$$

Consider also the ramping limits of the generator, which ensure that the unit operates within the permissible rates of change in production between consecutive periods.

$$(p_t + r_t) - p_{t-1} \leq RU \qquad \forall t \in \mathcal{T} \tag{2.20a}$$

$$-p_t + p_{t-1} \leq RD \qquad \forall t \in \mathcal{T}, \tag{2.20b}$$

where constraint (2.20a) ensures that the unit can provide spinning reserve within consecutive periods without violating the upward ramp limit.
In the SUC problem, the goal is to optimally schedule the generating units in order to maximize profits over the planning horizon. Consider, for instance, that there exists an agent who manages

a portfolio of generating units. The total revenue of the agent given by the difference between the revenue and the total operating costs:

$$\sum_{t\in\mathcal{T}}\sum_{g\in\mathcal{G}}\left[\pi_t\left(\underline{P}u_{g,t}+p_{g,t}\right)-c_{g,t}\left(u_{g,t},v_{g,t},w_{g,t},p_{g,t}\right)\right], \tag{2.21}$$

where π_t refers to the energy prices, and $c_{g,t}$ is the total operating cost per unit at each time period, which is given by:

$$\sum_{t\in\mathcal{T}}\sum_{g\in\mathcal{G}}\left[C_g^{NL}u_{g,t}+C_g^{LV}\left(\underline{P}u_{g,t}+p_{g,t}\right)+C_g^{SU}v_{g,t}\right] \tag{2.22}$$

The first term in equation (2.22) represents the no–load costs of the generator, which accounts for the cost of keeping a generator online regardless of its power production. The second term, stands for the linear variable cost, which depends directly on the amount of power the generator produces. Finally, the third term pertains to the start-up cost of the unit, representing the expenses incurred each time the generator is turned on. In this regard, the SUC problem can be formulated as follows:

$$\begin{aligned}\textbf{SUC:}\quad \max_{x,y}\quad & \sum_{g\in\mathcal{G}}\sum_{t\in\mathcal{T}}\left[\left(\pi_t\left(\underline{P}u_{g,t}+p_{g,t}\right)-c_{g,t}\left(u_{g,t},v_{g,t},w_{g,t},p_{g,t}\right)\right]\right.\\ \text{s.t:}\quad & (2.16),(2.17),(2.18),(2.20)\end{aligned} \tag{2.23}$$

Moreover, the UC problem is closely related to the SUC problem, differing primarily in their objective functions and operational context. Formally, the UC problem aims to minimize the total system operating cost, subject to the technical and operational constraints of the generating units, while ensuring that the total production satisfies system demand at each time period of the planning horizon. In the *3bin* UC formulation, the demand satisfaction constraints takes the following form:

$$\sum_{g\in\mathcal{G}}\left(\underline{P}_g u_{gt}+p_{gt}\right)=D_t\quad\forall t\in\mathcal{T}, \tag{2.24}$$

where, as in the formulation previously presented, D_t correspond to the demand of the system in time period t. In this case, it is worth noting that this constraint can also be extended to include non–served energy, enabling the model to account for situations in which demand is not fully satisfied and providing a mechanism to penalize or limit the load curtailment.

2.3 Polyhedral analysis

A polyhedron corresponds to a subset of $\mathbb{R}^n$ and can be defined as the solution set of a finite number of linear equalities and inequalities which can be written as:

$$\mathcal{P}=\left\{\boldsymbol{x}\mid a_j^T x\leq b_j,\ j=1,...,m,\ c_j^T x=d_j,\ j=1,...,p\right\} \tag{2.25}$$

According to the definition in (2.25) a given polyhedron $\mathcal{P}$ corresponds to the intersection of a finite number of halfspaces and hyperplanes, where a hyperplane is a subspace of $\mathbb{R}^n$ with dimension $n-1$ that divides the space into two half-spaces, and a half–plane is one of the halves after the hyperplane has been split into two.
The polyhedron $\mathcal{P}$ is defined by m inequality constraints and p equality constraints, whenever $p\geq 1$ then the equations of the form $c_j^T x=d_j$ define an

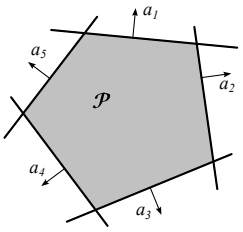


Figure 2.3: The polyhedron $\mathcal{P}$ is defined in $\mathbb{R}^2$ as the intersection of five hyperplanes with normal vectors a_j, for $j=1,..,5$, with, $p=0$.

affine subspace $\mathcal{V}$ of $\mathbb{R}^n$, of dimension $n-p$. For the given problem only the points $x \in \mathcal{V}$ are interesting. It would be possible to choose new coordinates y_k in such a way that $\mathcal{V} = \mathbb{R}^{n-p}$, for a more detailed description please refer to [55].

2.3.1 Integer programming

Integer programming corresponds to a class of mathematical optimization problems where some or all of the decision variables are restricted to take only integer values, and arise in practically every area of application of mathematical programming. In particular, suppose we have a linear program of the form:

$$\max \{\boldsymbol{cx} \ : \ \boldsymbol{Ax} \leq \boldsymbol{b}, \ \boldsymbol{x} \geq 0\}, \tag{2.26}$$

where $\boldsymbol{A}$ is an $m \times n$ matrix , $\boldsymbol{b}$ is an $m-$dimensional column vector, $\boldsymbol{c}$ is an $n-$dimensional row vector, (all involving parameters), and $\boldsymbol{x}$ is and $m-$dimensional column vector of variables. If we establish that some of the variables are integer we have mixed integer program (MIP), if all variables are integer we have an integer program (IP), if al variables are restricted to 0–1 values, what we have is 0–1 binary integer program.
It also exist combinatorial optimization problems, where given a finite set $N = \{1, ..., n\}$, weights c_j for each $j \in N$, and a set $\mathcal{F}$ of feasible subsets of N, we seek to find a minimum combination of weights over a certain subset of N. For further details about combinatorial optimization, please refer to [56, 57].

2.3.2 Linear programming relaxations

There are several types of relaxations, including linear, combinatorial, and Lagrangian relaxations; however, we will focus primarily on the linear type. Given an integer program such as $\max\{cx \ : \ x \in P \cap \mathbb{Z}^n\}$ where $P = \{x \in \mathbb{R}^n_+ \ : \ Ax \leq b\}$, the corresponding linear programming relaxation is given by $z^{\text{LP}} = \max\{cx \ : \ x \in P\}$. To illustrate the concept of relaxation we present the next example in two dimensions:

Example 2.1 Consider the following two dimensional integer programming problem given by:

$$\begin{aligned} z = \max \quad & 2x_1 + 2x_2 \\ \text{s.t.} \quad & 7x_1 - 2x_2 \ \leq 14 \\ & 2x_1 - 3x_2 \ \leq 3 \\ & 3x_1 + 21x_2 \leq 20 \\ & x \in \mathbb{Z}^2_+. \end{aligned}$$

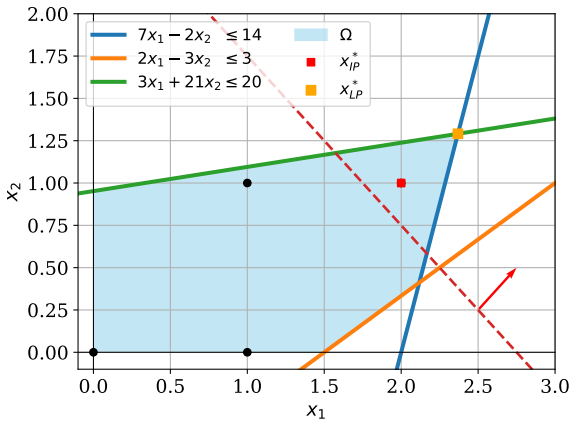


Figure 2.4: Feasible region (Ω) of Example 2.1.

Figure 2.4 illustrates the feasible region from Example 2.1, in which the orange and red squares represent the optimal solutions of the relaxed (x^*_{LP}) and integer (x^*_{IP}) problems, respectively. The red dotted line corresponds to an arbitrary level curve of the objective function along with the direction of optimization. As can be seen, the solution of the relaxed problem gives us an upper bound to the objective function of the integer problem, from the example we obtain $z \leq 7.32$. Observing that the optimal value must be an integer (because the objective function coefficients are also integers), and that we are maximizing, we can round down to the nearest integer and so obtain $z \leq 7$.

Observe that a good formulation is of great relevance given that they result in tighter bounds to the solution. The advantage of linear relaxation is that efficient algorithms exist for solving linear programs, which is of great importance in the context of large problems, contrarily to the case of mixed integer linear programs which are NP-hard [58].

2.3.3 Formulations of a polyhedron

In the context of integer programming, finding *good* formulations is a topic of great relevance [59], in order to accelerate the solution times to get good quality solutions.

Some of the efforts to accelerate the solution times involve relaxing the problem, i.e. drop integrality constraints on variables, which leads to a parameter called *integrality gap* that measures the difference between the objective value of the best feasible solution found so far and the current lower bound on the optimum.
In order to solve mixed or pure integer problems, the most common used strategy is the branch and bound (B&B) algorithm which relaxes the problem and solves disjoint small subproblems with fixed variables. In this context, providing effective relaxations is essential for the convergence of the B&B algorithm, as it determines the solution from which the algorithm begins branching.
To illustrate the concept of relaxation, lets took a 2–dimensional problem as shown in Figure 2.5.

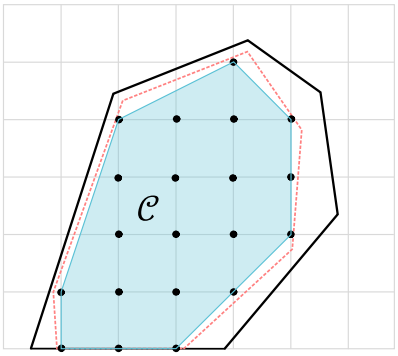


Figure 2.5: Three valid formulations $\mathcal{P}_1$ (black line), $\mathcal{P}_2$ (red dotted line), and $\mathcal{C}$ (blue colored one) for a pure integer programming set of points, where the the blue shaded formulation corresponds to the ideal formulation, i.e. the tightest one.

We say that the formulations $\mathcal{P}_1$ and $\mathcal{P}_2$ are equivalent if and only if they represent the same set of feasible solutions (points in the space).

$$\mathcal{S} = \{\boldsymbol{x} \in \mathbb{Z}^n \ : \ \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b},\ \boldsymbol{x} \geq 0\} \equiv \{\boldsymbol{x} \in \mathbb{Z}^n \ : \ \boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{d},\ \boldsymbol{x} \geq 0\} \tag{2.27}$$

Given two valid formulation of the same mixed–integer linear optimization problem, we say that formulation $\mathcal{P}_2$ is a stronger formulation with respect to $\mathcal{P}_1$ if

$$\{\boldsymbol{x} \in \mathbb{Z}^n \ : \ \boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{d},\ \boldsymbol{x} \geq 0\} \subset \{\boldsymbol{x} \in \mathbb{Z}^n \ : \ \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b},\ \boldsymbol{x} \geq 0\}$$

In other words, we say that the formulation $\mathcal{P}_2$ is tighter than $\mathcal{P}_1$, which means that all the points contained in $\mathcal{P}_2$ are strictly contained in $\mathcal{P}_1$, and therefore is a *better formulation*. Moreover, the idea of the convex–hull formulation can be formalized according to the definition in [56]:

Definition 1 *Given a set $X \subseteq \mathbb{R}^n$, the convex–hull of X, denoted conv(X), is defined as: conv(X) $= \{\boldsymbol{x} \mid \boldsymbol{x} = \sum_{i=1}^{t} \lambda_i x^i,\ \sum_{i=1}^{t} \lambda_i = 1,\ \lambda_i \geq 0$ for $i = 1, ..., t$ over all finite subsets $\{x^1, ..., x^t\}$ of $X\}$. conv(X) is a polyhedron, and all the extreme points of conv(X) lie in X.*

In particular, as can be deduced from Figure 2.5, if we solve a linear problem over $\mathcal{C}$, the optimal solution will lie in any extreme point of the polyhedron, which is also an integer point. The ideal formulation would allow us to solve the problem very efficiently, since the simplex method, or any algorithm that yields an optimal basic solution, would directly provide the optimal integer solution, eliminating the need for branching.
Example 2.1 illustrates the aforementioned concept by considering a finite set of integer points in a two-dimensional space. For this set, we introduce three alternative valid formulations, each of which characterizes the feasible region and provides a different polyhedral representation of

the same integer set. Although all formulations are equivalent in terms of their integer solutions, they differ in the strength of their linear programming relaxations, and thereby highlighting how alternative modeling choices can significantly affect the quality of a formulation.

Example 2.1 Consider a finite set of discrete points in the two-dimensional integer lattice $\mathbb{Z}^2$, denoted by:

$$\mathcal{P} = \{(2,1),(1,2),(1,4),(5,1),(4,5),(7,4)\},$$

and let $\mathcal{P}_1, \mathcal{P}_2$, and $\mathcal{P}_3$ represent three distinct valid formulations such that the set of integer solutions for each coincides exactly with $\mathcal{P}$. That is: $\mathcal{P} = \text{conv}(\mathcal{P}) \cap \mathbb{Z}^2 = \mathcal{P}_i \cap \mathbb{Z}^2 \quad \text{for } i \in \{1,2,3\}$.

$$\begin{aligned}
\mathcal{P}_1 =& \{\boldsymbol{x} \in \mathbb{R}^2 : -x_1 - x_2 \leq -3, -x_1 + 3x_2 \leq 11, \\
& \quad 3x_1 - 2x_2 \leq 13, x_1 + 3x_2 \leq 19\} \\
\mathcal{P}_2 =& \{\boldsymbol{x} \in \mathbb{R}^2 : 7/10 \leq x_1 \leq 36/5, 4/5 \leq x_2 \leq 36/5\} \\
\mathcal{P}_3 =& \{\boldsymbol{x} \in \mathbb{R}^2 : -60x_1 - 55x_2 \leq -152, \\
& \quad -110x_1 + 15x_2 \leq 47, 5x_2 \leq -4, \\
& \quad -2x_1 + 5x_2 \leq 19, 15x_1 - 5x_2 \leq 71, \\
& \quad 65x_1 - 105x_2 \leq 827\}
\end{aligned}$$

The following relation can be easily deduced by the figure on the right that $\mathcal{P}_1 \subset \mathcal{P}_2 \subset \mathcal{P}_3$. And they are equivalent with respect to integer points, in the sense that

$$\mathcal{P}_1 \cap \mathbb{Z}^2 \;=\; \mathcal{P}_2 \cap \mathbb{Z}^2 \;=\; \mathcal{P}_3 \cap \mathbb{Z}^2.$$

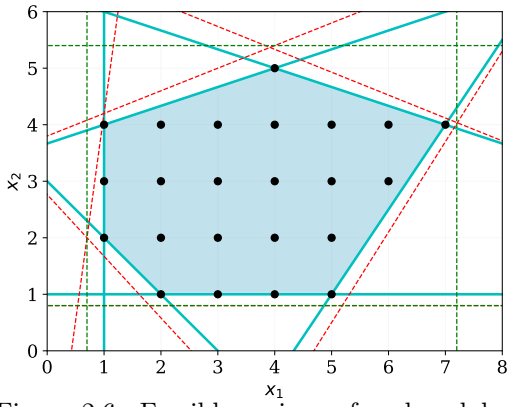


Figure 2.6: Feasible regions of each polyhedron, where the light blue, corresponds to $\mathcal{P}_1$, the red to $\mathcal{P}_2$ and the green to $\mathcal{P}_3$.

2.3.4 Faces

Given a non–empty polyhedron $\mathcal{P} = \{\boldsymbol{x} \in \mathbb{R}^n \;:\; \boldsymbol{A}x \leq \boldsymbol{b}\}$, i.e. the intersection of finitely many closed half–spaces. Also, let $\mathcal{F}$ be a face of $\mathcal{P}$ as any subset fo the form:

$$\mathcal{F} = \{\boldsymbol{x} \in \mathcal{P}^n \;:\; \boldsymbol{a}^T\boldsymbol{x} = b\}, \tag{2.28}$$

where $\boldsymbol{a}$ is a vector in $\mathbb{R}^n$, b a parameter in $\mathbb{R}$, and the linear inequality $\boldsymbol{a}^T\boldsymbol{x} \leq b$ is valid for the polyhedron $\mathcal{P}$. There are three distinct types of faces, each representing a specific case of a valid inequality. Note that faces do not give a minimal description of a polyhedron [60].
To build a better understanding, the three types of faces described above can be visualized in Figure 2.7 through a simple two–dimensional example. In this setting, vertices represent 0-dimensional faces, edges correspond to 1-dimensional faces, and the polytope itself illustrates the 2-dimensional face.
To prove that a valid inequality $\pi x \leq \pi_0$ is facet–defining for a given polyhedron X, one can find n points $x_1, ..., x_n \in X$, satisfying $\pi x = \pi_0$, and then prove that these n points are affinely independent. A set of points $p_0, ..., p_k \in \mathbb{R}^n$ is said to be affinley independent if and only if the only solution to

$$\sum_{i=0}^{k} \lambda_i p_i = 0, \quad \sum_{i=0}^{k} \lambda_i = 0,$$

is $\lambda_0 = \lambda_1 = ... = \lambda_k = 0$, which means that no point is an affine combination of the others. Consider the following propositions:

a. **0-dimensional face:** Corresponds in simple words to a single feasible point of the polyhedron. Formally, a face $\mathcal{F}$ of $\mathcal{P}$, satisfying $\dim(\mathcal{F}) = 0$, can be represented as $\mathcal{F} = \{v\}$, where $v \in \mathcal{P}$. Such a point v is called a vertex, or extreme point of $\mathcal{P}$.

b. **(n–1)-dimensional face:** A face $\mathcal{F} \subseteq \mathcal{P}$ of a polyhedron $\mathcal{P}$ is called proper if $\mathcal{F} \neq \emptyset$ and $\mathcal{F} \neq \mathcal{P}$. Among the proper faces, those of dimension $\dim(\mathcal{F}) = n - 1$, where $n = \dim(\mathcal{P})$, are called facets. Equivalently, a facet is a maximal proper face of the polyhedron $\mathcal{P}$.

c. **Empty face:** An empty face corresponds to any valid inequality of the form $\boldsymbol{a}^T\boldsymbol{x} \leq b$ for which there does not exist any $\boldsymbol{x} \in \mathcal{P}$ such that $\boldsymbol{a}^T\boldsymbol{x} = b$.

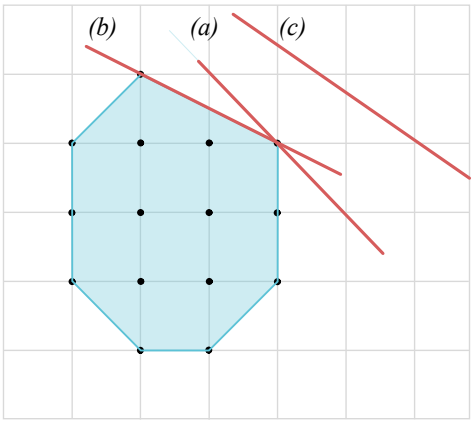


Figure 2.7: 2-dimensional polyhedron with three types of faces, where (a) represents a facet of the polyhedron, (b) denotes a 0–dimensional face, and (c) corresponds to an empty face.

Proposition 1 *Points $p_0, ..., p_k \in \mathbb{R}^n$ are affinely independent if and only the k vectors*

$$v_1 = p_1 - p_0, \quad v_2 = p_2 - p_0, \quad ..., \quad v_k = p_k - p_0,$$

are linearly independent in $\mathbb{R}^n$.

Example 2.2 Take for instance the following four point in $\mathbb{R}^4$:

$$p_0 = (0,0,0,0), \quad p_1 = (1,0,0,0), \quad p_2 = (0,1,0,0), \quad p_3 = (0,0,1,0)$$

Form the difference vectors v_i based at p_0:

$$v_1 = p_1 - p_0 = (1,0,0,0), \quad v_2 = p_2 - p_0 = (0,1,0,0), \quad v_3 = p_3 - p_0 = (0,0,1,0)$$

These three vectors are are linearly independent (no nontrivial linear combination gives the zero vector). By the characterization in the proof you already saw, this implies p_0, p_1, p_2, p_3 are affinely independent. Note that, the largest possible affinely independent set has size $n + 1$. Considering the given example, if we add $p_4 = (0,0,0,1)$ yields five affinely independent point.

2.3.5 Cutting planes

Cutting planes is a fundamental technique in optimization, particularly in solving integer programming problems [61, 51, 62]. They are a method to tighten the feasible region of a linear programming (LP) relaxation in mixed-integer programming (MIP) problems. In optimization, many problems can be modeled as integer or mixed-integer programs, where some or all of the decision variables are constrained to be integers [63]. Solving these problems is challenging due to the combinatorial nature of integer constraints. Cutting plane methods aim to improve the efficiency of solving these problems by iteratively adding linear inequalities, called cuts, to the LP relaxation [64]. A mixed integer linear programming problem can be expressed in the following compact form:

$$\max \left\{ \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{y} : (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{S} \right\} \tag{2.29}$$

The variables x_j are constrained to be non–negative integers while the variables y_j are free to take any non–negative real value. In particular, $\mathcal{S} := \left\{ (\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{Z}^n_+ \times \mathbb{R}^p_+ \ : \ A\boldsymbol{x} + G\boldsymbol{y} \leq b \right\}$ corresponds to the feasible set of the variables called *mixed integer set*, a pure integer linear program is the

special case of the mixed integer linear program where $p = 0$. Let also (x^*, y^*) be an optimal solution, and let z^* be the optimal value for the MILP solution.
Furthermore, consider a natural linear programming relaxation of the MILP problem obtained by discarding the integrality requirement on the vector x. The mentioned optimization problem can be expressed as:

$$\max \left\{ \boldsymbol{c^T x} + \boldsymbol{d^T y} : (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{P}_0 \right\} \tag{2.30}$$

where $\mathcal{P}_0 := \left\{ (\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^n_+ \times \mathbb{R}^p_+ \ : \ A\boldsymbol{x} + G\boldsymbol{y} \leq b \right\}$, consider the optimal solution for the relaxed problem, denoted as (x^0, y^0), and the optimal value of the natural linear programming relaxation, called z^0. We assume that (x^0, y^0) is a basic optimal solution of (2.30), which is available to us, as well as z^0. Since $\mathcal{S} \subseteq \mathcal{P}_0$, we can say that z^* is a lower bound on z^0, namely $z^* \leq z^0$. There exist two possibilities when we find the optimal solution to the relaxed problem, which can be represented on the flow chart in the Figure 2.8. The linear constraints added to each iteration are the so-called cuts, which wish to tighten the space by cutting fractional solutions. Also called valid inequalities, they can be expressed as $\alpha\boldsymbol{x} + \gamma\boldsymbol{y} \leq \beta$.
These inequalities satisfy the integer constraints of the original problem and help define a tighter feasible region without excluding any potential integer solutions [65]. The defined cutting plane forms a new set called $\mathcal{P}_1$ wish can be expressed as:

$$\mathcal{P}_1 := \mathcal{P}_0 \cap \{(\boldsymbol{x}, \boldsymbol{y}) : \alpha\boldsymbol{x} + \gamma\boldsymbol{y} \leq \beta\} \tag{2.31}$$

This last set is tighter than $\mathcal{P}_0$, so the optimal value of the objective function with $\boldsymbol{x}$ and $\boldsymbol{y}$ lying on $\mathcal{P}_1$ is at least as good upper–bound on z^* as z^0.
For an arbitrary integer programming problem, consider the three optimization problem, which describe the same feasible set of points.

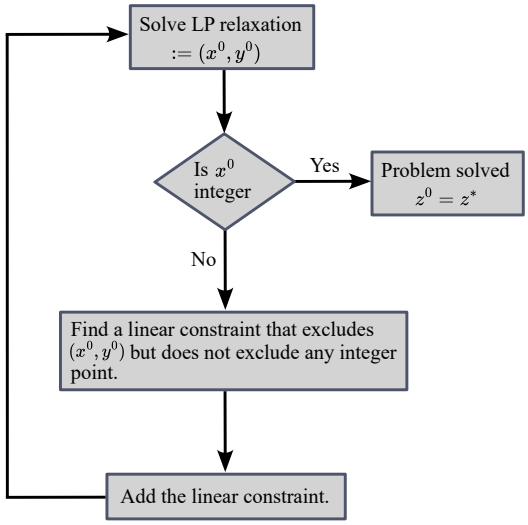


Figure 2.8: Flow chart for the cutting plane algorithm, considering two types of variables, y required to be real while x constrained to be integer.

Table 2.1 summarizes the three common formulations of an integer optimization problem. The first column presents the IP, where the decision variables x are constrained to take integer values. The second column shows the corresponding LP relaxation, in which the integrality constraints are removed; this relaxation often provides a bound on the optimal value of the integer program. The third column illustrates the LP relaxation strengthened with cuts ($\alpha x \leq \beta$), which serve to tighten the relaxation without excluding any feasible integer solutions.

Integer Program:	**LP Relaxation:**	**LP Relaxation + Cuts:**
min $c^T x$	min $c^T x$	min $c^T x$,
s.t: $Ax \geq b$,	s.t: $Ax \geq b$,	s.t: $Ax \geq b$,
$x \geq 0$,	$x \geq 0$.	$x \geq 0$,
x integer.		$\alpha x \leq \beta$.

Table 2.1: Comparison of an integer program, its linear programming relaxation, and the LP relaxation strengthened with valid inequalities (cuts).

Based on the three optimization problems presented above, Figure 2.9 illustrates the LP Relaxation

on the left and the LP Relaxation tightened with cutting planes on the right, through a two–dimensional representation. The cutting plane, highlighted in red, is represented by the inequality $\alpha x \leq \beta$.

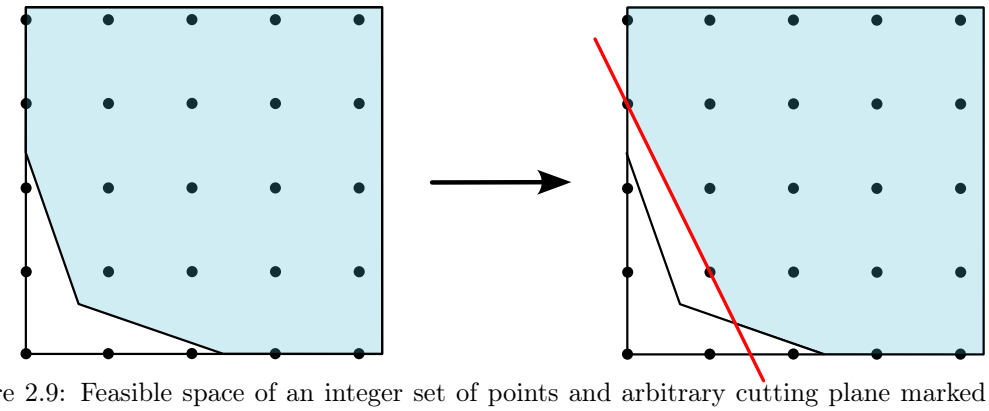


Figure 2.9: Feasible space of an integer set of points and arbitrary cutting plane marked in red, where the set on the left correspond to the LP Relaxation of the IP, while the set on the right to the LP Relaxation with a cutting plane.

Moreover, there have been research effort focused toward deriving facets for some specific structures. In this way, the polyhedral study of the knapsack problem has been fundamental in developing efficient cutting planes for integer programming. Early research studied the structure of the classic 0/1 knapsack polytope, by E. Balas in 1975 [66] introducing a significant lifting procedure running in linear–time for cover inequalities that avoids the computational cost of solving auxiliary optimization problems. Simultaneously, L. Wolsey [67] identified necessary and sufficient conditions under which 0–1 inequalities define faces of the associated polyhedron, introducing the concept of *strong covers*. The theory was later expanded to include more complex structures, such as generalized upper bound constraints. For instance, also L. Wolsey in 1990 [68] investigated the knapsacks set with both positive and negative coefficients, introducing *GUB cover* inequalities. In the context of mixed 0–1 regions, [69] and [70] provided extensive analyses of generalized flow cover inequalities, focusing on facet–defining conditions, separation problems, and numerical experiments with lifting. Also, the knapsack problem have been studied in the context of continuous variable [71], as well as fixed–charge problems [72]. More recently, attention has turned to semi–continuous knapsack problems. This includes studies on the convex–hull of sets where variables belong to unions of intervals [73], and the investigation of semi–continuous flow variables in network problems with GUB and demand constraints [74, 75].

Consider for instance a two–dimensional two variables, x_1 and x_2, pure integer programming program. Figure 2.10 shows two different valid formulations of $\mathcal{X}$ and an example of a cutting plane that cuts fractional solutions that doesn't belong to conv($\mathcal{X}$). The tighter formulation corresponds to the convex hull of $\mathcal{X}$, which is the ideal formulation for any integer programming (IP). On the other hand, $\mathcal{P}_1$ corresponds to a particular formulation for $\mathcal{X}$ among infinite possible formulations. A wide range of cutting–plane methods has been developed to strengthen the formulations and improve solution efficiency. Among these are the Gomory mixed–integer cuts, which exploit the fractional components of linear programming relaxations to iteratively tighten the feasible region [76].

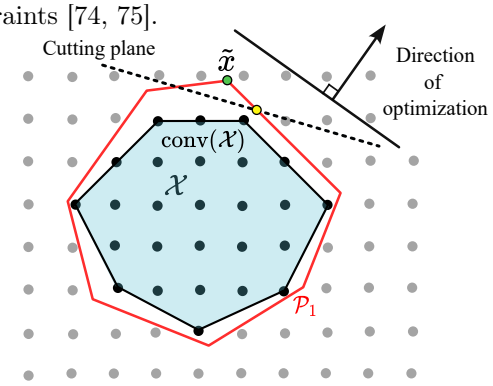


Figure 2.10: Convex hull of the integer set $\mathcal{X}$ (black dots), with optimization direction (solid line) and a cutting plane (dotted line).

2.3.6 Branch and bound

The branch–and–bound (B&B) procedure, also known as *divide–and–conquer*, is a very widely used method to solve MILP optimization problems which was first proposed in [77]. The main idea of the algorithm is to successively divide the given problem into smaller subproblem that carry information of past iterations, in order to get an optimal solution among all subproblems.
Define an optimization problem $\mathcal{P} = (X,\ f)$, where X corresponds to the set of all feasible solutions of the optimization problem $\mathcal{P}$, and f the corresponding objective function. To solve $\mathcal{P}$, the B&B algorithm builds a search tree where relaxed subproblems of $\mathcal{P}$ are solved in each node of the tree, giving for each subproblem $\mathcal{SP}$, a feasible solution x^* which si globally stored. If a solution $x^{*\prime}$ of a subproblem $\mathcal{SP}'$ is found, which is better than (i.e. $f(x^{*\prime}) < f(x^*)$ in a minimization context) we call this solution an incumbent of the search tree, and the incumbent value is updated. Contrarily, if no other solution better than x^* can be found among all the nodes of the subtree, the former is pruned and the algorithm moves towards another unexplored subtree. Once no unexplored subtrees remain, the best incumbent solution x^* called global optimum is returned, and the resulting solution satisfies $x^* \in \operatorname{argmin}_{x \in X} f(x)$. Pseudocode for the generic B&B procedure is given in Algorithm 1 [65]. For a more detailed survey of the B&B algorithm please refer to [78, 79].

Algorithm 1 Branch–and–Bound: Minimization problem instance R.

Require:
1: $\mathcal{L} \leftarrow \{R\}$ ▷ initialize
2: $\hat{c} \leftarrow \infty$ ▷ initialize
3: **while** $\mathcal{L} \neq \emptyset$ **do**
4: Choose $Q \in \mathcal{L}$ ▷ select node
5: $\mathcal{L} \leftarrow \mathcal{L} \setminus \{Q\}$ ▷ actualize
6: Solve a relaxation Q_{relax} of Q ▷ solve
7: **if** $Q_{\text{relax}} := \emptyset$ **then** $\check{c} \leftarrow \infty$ ▷ assignment
8: **else** $\check{x} \leftarrow$ Optimal solution for Q_{relax}, $\check{c} \leftarrow$ Objective value of Q_{relax}. ▷ assignment
9: **end if**
10: **if** $\check{c} \geq \hat{c}$ **then goto** 3. ▷ bound
11: **end if**
12: **if** $\check{x}$ is feasible for R **then** $\hat{x} \leftarrow \check{x}$, $\hat{c} \leftarrow \check{c}$, **goto** 3. ▷ check
13: **end if**
14: Split Q into subproblems $Q_1, \ldots, Q_k$ ▷ branching
15: $\mathcal{L} \leftarrow \mathcal{L} \cup \{Q_1, \ldots, Q_k\}$ ▷ branching
16: **end while**
17: **if** $c^* = \hat{c} := \infty$ **then**
18: **return** No optimal solution found. ▷ report solution
19: **else**
20: **return** Optimal solution found $x^* = \hat{x}$ with value $c^* = \hat{c}$. ▷ report solution
21: **end if**

Figure 2.11 illustrates the structure of a B&B search tree used to solve a mixed–integer optimization problem. The root node at the top represents the LP relaxation of the original problem, where integrality constraints are relaxed [80]. Solving this relaxation provides an initial bound on the optimal objective value. If the solution at the root node is integer–feasible, it is optimal and the algorithm terminates, in such case, the given formulation corresponds to the convex–hull of the formulation. When the solution at a node violates integrality requirements, the algorithm branches by selecting a fractional variable and creating child nodes, each corresponding to additional constraints that partition the feasible region. Each node in the tree therefore represents a subproblem with a more restricted feasible set.

Consider for instance a minimization problem were the root node provides the first lower bound for the objective function. As the algorithm starts branching, subproblems are being solved and the lower and upper bound are respectively updated.
Nodes are pruned when it can be established that exploring its descendants cannot lead to an improvement over the current best solution. The main reasons for pruning a node of the tree are as follows. First, bound–based pruning occurs when the bound obtained from solving the LP relaxation at a node is dominated by the incumbent objective value. In the case of a minimization problem, if the lower bound of the node is greater than or equal to the current global upper bound, the corresponding subproblem cannot contain an optimal solution and is therefore discarded.

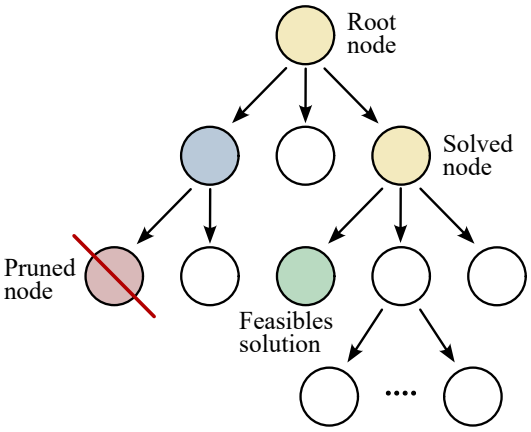


Figure 2.11: Illustration of the branch and bound search tree, in which yellow nodes denote solved subproblems, the green node indicates a feasible solution, and red nodes correspond to subproblems that have been pruned.

Second, infeasibility pruning is applied when the LP relaxation of a node is infeasible, indicating that the additional constraints introduced by branching make respective the subproblem inconsistent. In this case, no feasible solution exists in the corresponding region of the search space, and therefore is discarded.
Finally, nodes may also be pruned because of the optimality gap tolerance which is a pre–defined criterion, where a node is discarded if the difference between its bound and the incumbent is lower than, or equal to the numerical tolerance. This criterion ensures finite termination and numerical stability in practical implementations of the algorithm.

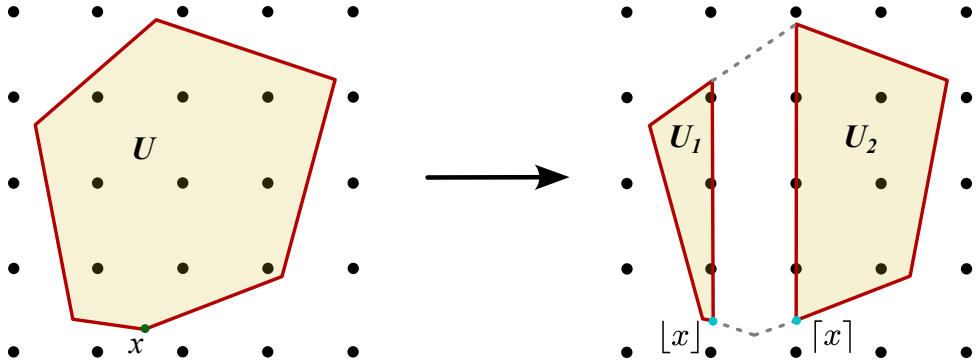


Figure 2.12: Two-dimensional illustration of branching on a single fractional variable, resulting in two disjoint regions.

Figure 2.12 illustrates the branching process on a fractional variable x, resulting in two disjoint sets U_1, and U_2. In order to divide the set U, constraints $x \leq \lfloor x \rfloor$ and $x \leq \lceil x \rceil$ are added to the problem. Specifically the set U_1 can be written as $U \cap \{x \leq \lfloor x \rfloor\}$, while set U_2 can be written as $U \cap \{x \leq \lceil x \rceil\}$.
Furthermore, the branch–and–cut algorithm combines the B&B techniques along with the cutting planes method, thereby improving the performance of the discussed algorithm [81]. At each node, instead of solving only the original LP relaxation, the formulation is iteratively tightened by adding cutting planes that remove fractional solutions without excluding any integer–feasible solutions. These cuts improve the quality of the node bounds, often leading to faster bound closure and increased pruning of the search tree.

2.3.7 Tightness vs. compactness

As demonstrated previously, multiple valid formulations may exist for a single MILP instance. While each formulation ultimately yields an identical optimal solution, they may differ significantly in computational tractability and convergence behavior. Two main properties governing solver performance are tightness–the proximity of the LP relaxation to the convex–hull of the feasible integer set–and compactness, which refers to the polynomial relationship between the number of variables/constraints and the problem size. During the solving process of a MILP problem using the B&B techniques, upper and lower bounds to the objective function are computed in every feasible. In the context of these quantities, presolve and heuristics are implemented by the solver to improve the upper bound, while cutting planes contribute in strengthen the lower bound, to ultimately close the gap between both bounds [82, 83].

In simple terms, presolve refers to the process of transforming a problem P into an equivalent problem P' that can be solved more efficiently by the chosen solver, generally the problem P' is at least more compact (because redundant variables and constraints are removed), and tighter (because constraints and variable bounds are strengthened) than the original problem P. Presolve routines comprise a suite of preprocessing techniques designed to reduce the problem instance by identifying and eliminating redundant constraints, fixed variables, and linearly dependent rows. By tightening variable bounds and performing coefficient reduction, these procedures refine the polyhedral description of the model [84]. Consequently, presolving reduces the computational burden of the subsequent search phase, often leading to a more tractable LP relaxation and a more efficient exploration of the branch–and–bound tree. For some cases, node presolve can make the difference between an intractable instance and a solvable one [85].

Primal heuristics represent a significant advancement in the state of the art for improving the computational performance of MILP solvers. Although they are not strictly necessary to guarantee correctness, they are widely implemented across modern solvers because they substantially accelerate the search for high-quality feasible solutions. Heuristics are executed at specific nodes of the branch–and–bound tree with the aim of improving the corresponding objective value, working as a complement of the branching process. Two main classes of heuristics are commonly employed by MILP solvers. The first class corresponds to starting heuristics, which aim to improve the solution quality at the root node of the algorithm. And the second class correspond to the so called improvement heuristics, aiming to improve the solution in the nodes of the tree during the branching process.

Cutting–plane techniques lie at the core of efforts to improve the tightness of MILP formulations. State of the art solvers employ a wide variety of cutting–plane families to strengthen the relaxation, including Knapsack covers, GUB covers, FLow covers, Cliques, Implied bounds, and Gomory mixed–integer cuts. Cutting planes introduce valid inequalities that eliminate fractional solutions of the linear programming relaxation while preserving all feasible integer solutions. Their primary effect on the underlying polyhedron is to tighten the relaxation, thereby bringing it closer to the convex hull of feasible integer points. As a consequence, stronger dual bounds are obtained at earlier stages of the solution process, which in turn reduces the need for extensive branching in the search tree.

As discussed in the previous paragraphs, there is a clear trade–off between tightness and compactness in MILP formulations. In particular, a formulation can be tightened by adding additional constraints, but this typically harms compactness by increasing the number of rows and columns. For this reason, the presolve phase plays a crucial role by identifying and removing redundant or unnecessary elements of the formulation, thereby reducing its size while preserving its strength. Conversely, attempts to make a formulation more compact, such as eliminating constraints, often weaken its tightness and enlarge the integrality gap.

Consider formulations F_1 and F_2 for the same optimization problem as shown in Figure 2.13. Consider that the formulation F_2 contains the feasible region of formulation F_1, that is, $F_1 \subseteq F_2$ meaning that for every optimization direction, the formulation F_1yields a tighter bound, and therefore a smaller integrality gap, than the LP relaxation of F_2. In addition, F_1 achieves this tighter relaxation using a comparable or smaller number of variables and constraints, then F_1 is also said to be more compact. A formulation that is both tighter and more compact is generally preferable, as it yields stronger bounds without incurring harming computational performance.

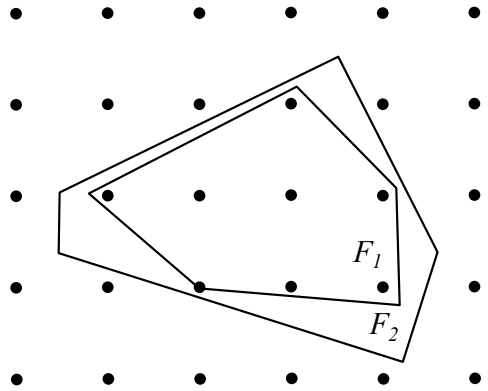


Figure 2.13: Formulations F_1 and F_2 for the same integer set.

This trade–off highlights the importance of valid inequalities because they strengthen the formulation by tightening the relaxation without being part of the original model. Modern solvers also provide mechanisms for incorporating user–defined valid inequalities, such as the use of *lazy constraints*, which allow additional cuts to be generated dynamically during the solution process, and added only if necessary.

Chapter 3

Mathematical formulations

This chapter presents the mathematical formulation underlying the main contributions of this thesis, together with a motivating example that illustrates the driving ideas of the work and highlights its relevance within the considered context. Subsequently, the polyhedron under study is formally introduced and rigorously described, along with the assumptions adopted to derive the ensuing polyhedral results. Next, the methodological procedure employed to carry out the polyhedral analysis is outlined, and the analytical tools used throughout this process are discussed. Furthermore, the polyhedral analysis is initiated by presenting a set of fundamental (trivial) proofs, which constitute a standard preliminary step in the derivation of valid inequalities and facet-defining conditions for a polyhedron. Finally, once the polyhedron is shown to possess the desired structural properties, the resulting cutting planes are introduced, together with the conditions under which they define facets, or alternatively, remain merely valid inequalities.

3.1 Motivational example

To provide intuition for the problem studied in this thesis work, we propose a simple self–UC setup composed by a single generating as depicted in Figure 3.1. In this illustrative framework, the unit determines its commitment and dispatch schedule based on the marginal price at its connection bus. Despite the simplicity of the framework, the proposed setup captures the essential features of the problem addressed in this thesis.

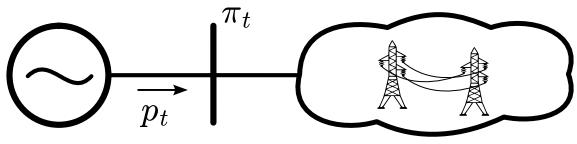


Figure 3.1: Illustrative example of a single conventional generating unit connected to a power system bus, where the scheduling decisions of the unit are driven by the marginal prices π_t at that bus.

As discussed in the previous section, the scheduling problem of a generating unit can be formulated in several alternative ways. In this work, we focus on two widely used mixed–integer formulations: the network–flow formulation and the *3bin* formulation. Both formulations explicitly model the unit commitment and dispatch decisions of the generator, capturing its operational constraints and cost structure. Although their mathematical representations differ, they are theoretically equivalent in the sense that they define the same feasible set in the space of commitment and

production variables. Consequently, for a given set of input parameters, both formulations yield identical optimal schedules for the generating unit and attain the same optimal objective function value.
In order to explicitly motivate and justify the scope of this work, we formulate the self–UC problem both with and without ramping constraints. This comparative modeling framework allows us to highlight the structural impact of ramping constraints by examining the resulting differences in the integrality gap of the corresponding mixed–integer formulations. In particular, this comparison provides insight into how ramping constraints affect the tightness of the linear programming relaxation and, consequently, the computational difficulty of the problem, and therefore why the polyhedral analysis of the ramping polytope of the network–flow formulation is relevant in this context.
In the same context, we present the results of the *3bin* model in order to benchmark the standard network flow formulation performance. For the given case of study the marginal prices in the connected bus are presented in the Table 3.1.

$t = 1...12$	$\rightarrow$	13.0	7.2	4.6	3.3	3.9	5.9	9.8	15.0	22.1	31.3	33.2	24.8
$t = 13...24$	$\rightarrow$	19.5	16.3	14.3	13.7	15.0	17.6	20.2	29.3	49.5	53.4	30.0	20.2

Table 3.1: Energy prices ($/MWh)

Furthermore, consider a generating unit with a maximum and minimum power output of 358 MW and 134 MW, respectively, and a ramping capacity of 28 MW/h, also, the units shut–down and start–up capacity are equal to the minimum power output. The unit is subject to a minimum up–time of four periods and a minimum down–time of two periods. As can be seen in Figure 3.2, the graph $\mathcal{G}$ of the unit is vertically composed by 6 nodes.
Where nodes from $E1$ to $E4$ correspond to *on* nodes, whereas nodes from $E5$ to $E6$ represent states in which the unit is *off*. Transition within nodes in row $E1$ represent operating states in which the unit is *on* and available for shut–down. In addition, transitions from row $E1$ to row $E4$ capture the change in operating status from *on* to *off*. Similarly, transitions from row $E2$ to $E1$ and from row $E3$ to $E2$ represent operating states in which the unit remains *on* and is awaiting the completion of its minimum up-time requirement. In the same way, transitions from row $E4$ to $E5$ and from row $E5$ to $E6$ represent operating states in which the unit is *off* and is awaiting the completion of its minimum down-time requirement. Finally, transition within nodes in row $E6$ represent operating states in which the unit is *off* and available for start–up.
Moreover, in this particular example, an injection of one unit of flow at node $E2$ indicates that the generating unit has been in the *on* state for two consecutive periods and must remain operational for two additional periods before a shut–down becomes feasible.
We present the following results to demonstrate the necessity of an enhanced network–flow formulation, as well as to highlight the impact of ramping constraints on both formulations. In particular, these results aim to quantify how gradient constraints influence the tightness of the corresponding relaxations and the overall polyhedral structure. The motivating experiment compares the integer programming formulation with its corresponding linear programming relaxation, both with and without ramping constraints, to evaluate how ramping constraints influence the resulting integrality gap.
To begin, we study a first relaxation of both formulations obtained by omitting the gradient constraints. Under this relaxation, the generating unit is allowed to adjust its production level within its admissible bounds between its minimum and maximum power outputs across consecutive periods, without being subject to intertemporal constraints, in order to maximize its profits.
As depicted in Table 3.2, both the NF and the *3bin* formulation describe the convex–hull of the integer points for the given operational polyhedron. The authors in [8] proved that self–UC problem, when described solely by the basic operating constraints, precisely characterizes the

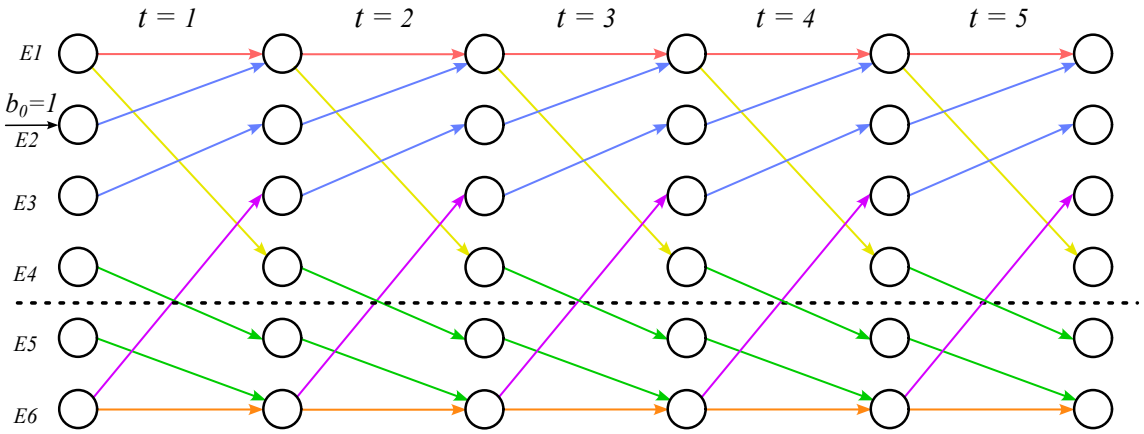


Figure 3.2: Network representation of the proposed example. Six distinct types of edges can be identified, ordered from top to bottom: red edges correspond to on – ready to shut–down transitions; yellow edges represent on – shut–down transitions; blue edges denote on – waiting for minimum up time states; purple edges correspond to off – start–up transitions; green edges represent on – waiting for minimum down time states; and orange edges denote off – ready to start up states.

Days	IntGap (%)		Sol. Time (s)	
	NF	***3bin***	***NF***	***3bin***
64	0.0	0.0	0.14	0.11
256	0.0	0.0	1.02	0.42

Table 3.2: Evaluation of both relaxations with respect to their integrality gaps and corresponding solution times for different horizons.

convex–hull of the corresponding polyhedron, which is consistent with the results shown in the table above, for more details regarding the convex–hull proofs, please refer to the related work.
Furthermore, since the NF constraints, define a polyhedron whose matrix of parameters A is TU [51, 61] and the right–hand side vector b consists of integers, every vertex (or basic feasible solution) of the feasible polyhedron defined by constraints is guaranteed to be integer, which means that the solution of the relaxed–problem exactly matches the solution of the original problem, making the problem very easy to solve with standard techniques. A matrix A is defined to be TU if every square submatrix of A has a determinant equal to 0, $+1$, or -1. See chapters 19 to 21 from [86] for more details about total unimodularity and its corresponding applications.
Nevertheless, adding ramping constraints to the nodes in the networks changes the nature of the matrix A, resulting in a constraint matrix whose structure no longer preserves total unimodularity and thereby alters the polyhedral properties of the feasible region. Consequently, the corresponding results for both formulations with ramping constraints are presented in Table 3.3.
As illustrated in Table 3.3, the inclusion of gradient constraints modifies the polyhedral structure of the formulation, impacting the properties observed in the relaxed model. As expected, the computational effort required to solve the formulations increases substantially relative to the results reported in the preceding table. Furthermore, considering the integrality gap results, it can be observed that the *3bin* formulation with ramping constraints is tighter than the network flow formulation with ramps; in particular, the integrality gap of the standard NF formulation is approximately twice that of the *3bin* formulation.
The polyhedral properties of the *3bin* formulation have been extensively studied in the litera-

Days	IntGap (%)		Sol. Time (s)	
	NF	***3bin***	***NF***	***3bin***
64	40.97	22.37	0.97	0.75
256	41.12	22.56	8.12	16.10

Table 3.3: Evaluation of te quality of the formulation with ramps with respect to their integrality gaps and corresponding solution times for different horizons.

ture, with significant research effort directed toward obtaining stronger and tighter formulations. In particular, valid inequalities have been derived for the ramping polytope [2], more compact and tighter formulations have been proposed for the complete unit commitment problem [22], convex–hull descriptions have been established for relaxations of the former formulations [8], and particular structures have been studied to derive strong valid inequalities [30, 87], as well as other contributions in the literature.
Nevertheless, comparatively little effort in the state of the art has been directed toward deriving tighter formulations for the NF–based formulation. Given the significant impact of ramping constraints on the quality of the formulation, in the following section we present a polyhedral study of the ramping polytope, aimed at deriving strong valid inequalities that, in specific scenarios, define facets of the polyhedron.

3.2 Problem statement

We consider the semi–continuous knapsack set with generalized upper–bound (GUB) constraints defined in (3.1). The set describes the feasible region induced by a single knapsack–type capacity constraint and two GUB constraints associated with disjoint subsets of items. Specifically, the ground set N is partitioned into two disjoint groups, $\mathcal{M}$ and $N \setminus \mathcal{M}$, from which at most one item can be selected in each group. The binary variables x_k indicate whether item k is selected, while the continuous variables y_k represent the corresponding semi–continuous activity levels, which are restricted to be zero whenever $x_k = 0$.
This structure captures the semi–continuous nature of the set while enforcing mutually exclusive selection within each disjoint group.

$$\mathcal{P} = \left\{ \begin{array}{lrcll} (x,y) \in \{0,1\}^n \times [0,1]^n : & \sum_{k\in N}(a_k x_k + m_k y_k) & \leq & \Delta & \\ & \sum_{k\in\mathcal{M}} x_k & \leq & 1 & \\ & \sum_{k\in N\setminus\mathcal{M}} x_k & \leq & 1 & \\ & y_k & \leq & x_k & k \in N \end{array} \right\}, \tag{3.1}$$

where $\mathcal{M}$ is an arbitrary set defined by $\{k \in N : \text{sign}\,(a_k)\}$, i.e. the parameters a_k that have equal sign, and $\mathcal{M} \neq \emptyset$, also let n be a positive integer such that $N = \{1, ..., n\}$ and a_k, m_k and Δ parameters in $\mathbb{R}$. Note that $a_k x_k + m_k y_k$ is a model for a semi–continuous variable with values in $\{0\} \cup [a_k, a_k + m_k]$. In general terms, a variable is referred to as semi–continuous if it is constrained to take values in a set of the form $\{0\} \cup [\ell,\, u]$, where $0 \leq \ell \leq u$. The parameters ℓ and u are respectively denoted as the lower and upper bounds of the variable when it is active. The first constraint is the semi–continuous knapsack constraint, the second and third constraints imposes a generalized upper bound (GUB) condition among disjoint sets of binary variables and the last inequality ensures semi-continuity.
Note that the set $\mathcal{P}$ can be viewed as the feasible region of a single–node gradient-constrained network, as illustrated in Fig. 3.3. In this representation, x_k is a binary variable that indicates whether the arc is utilized, while y_k denotes the flow through the arc above a lower bound, which

upper bound is limited by the binary variable. Additionally, the coefficients a_k and m_k represent the fixed and variable capacities of the arc k, respectively.

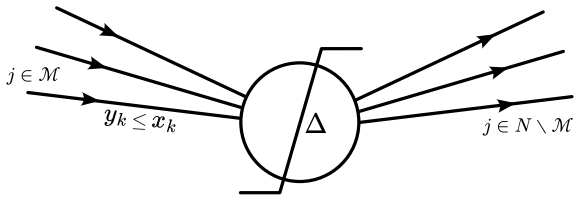


Figure 3.3: Single node flow model with gradient constraint, where, according to the direction of the flow, the right–hand side corresponds to the a_k's and m_k's being greater than zero, while the left–hand side involves the a_k's and m_k's being less than zero. Set $\mathcal{M}$ was chosen for this illustrative example such that; $\operatorname{sign} a_k > 0$.

3.3 Proposed framework of study

In this section, we present the proposed framework used to characterize the polyhedron $\mathcal{P}$ presented in (3.1) Specifically, we seek to identify the set of extreme points in order to derive the hyperplane characterization of the former which correspond to the base of the polyhedral study. The methodology applied consist of two relevant steps:

1. The first step consist on the development of an algorithm capable of identifying all extreme points of a given instance of the polyhedron. Specifically, given the instance parameters: $\boldsymbol{a}$, $\boldsymbol{m}$, δ, and n the algorithm computes the set of extreme points, denoted by $\mathcal{V}$.

2. The second step consists of obtaining the hyperplane representation of the specific instance of the polyhedron, using the set of extreme points computed in Step 1. This is accomplished through the use of the software PORTA [21], which is specifically designed for polyhedral analysis. In particular, the function *.traf* allows the user to input an arbitrary set of points and obtain the convex hull of these points, thereby generating the corresponding hyperplane representation. Notably, this procedure is reversible, allowing conversion from a hyperplane representation back to the set of extreme points if needed. The function, mentioned function *.traf* uses a Fourier–Motzkin elimination algorithm [88] which projects a linear system on subspaces $x_i = 0$.

3.3.1 Extreme point representation

As previously mentioned, the first step of our study consist in determining the set of extreme points for a given instances. An instance is defined by arbitrary matrices $\boldsymbol{a}$ and $\boldsymbol{b}$, along with a gradient parameter δ, and the given dimension of the polyhedron equal to $2n$, where $n = \dim(\boldsymbol{a}) = \dim(\boldsymbol{b})$. In order to determine the set of extreme points, a compact representation of the methodology is presented Algorithm 2. In this context, the input matrix $\boldsymbol{A}$, is defined as $\boldsymbol{A} = (\boldsymbol{a}, \boldsymbol{m})$, and the right–hand–side of the inequality denoted as b correspond to the gradient parameter δ.
Consider the polyhedron $\mathcal{P}$ defined as $Ax^T \leq b$, where $A \in \mathbb{R}_+^{m \times n}$ is the constraint matrix (or left–hand–side), $b \in \mathbb{R}_+^m$ is the right-hand side vector, n denotes the dimension of the space, and m represents the number of hyperplanes supporting the polyhedron. In this context, a straightforward approach to determine an arbitrary vertex of the polyhedron is to select a number equal to n inequalities of the form $a_i x^T \leq b_i$, where a_i and b_i denote the i-th row of matrices A and b respectively [89]. The corresponding vertex is then obtained by solving the resulting $n \times n$ system of linear inequalities, which can be written in compact form as $A'x^T \leq b'$, where $A' \in \mathbb{R}_+^{n \times n}$ and $b \in \mathbb{R}_+^n$. When solving the system, three distinct cases can be distinguished [90, 91]:

a. The system has no solution, which geometrically means, that the selected n hyperplanes do not intersect at a single point in $\mathbb{R}^n$, so no point simultaneously satisfies all n equations.

b. The system has no unique solution, which geometrically means, that the selected n hyperplanes intersect along a line, a plane, or a higher–dimensional subspace in $\mathbb{R}^n$ rather than at a single point. In this situation, there are infinitely many points that satisfy the system $A'x^T \leq b'$.

c. The system has a unique solution, which geometrically means that the selected subset of inequalities intersect at a single point in $\mathbb{R}^n$, and this point satisfies all inequalities defining the polyhedron $\mathcal{P}$.

Algorithm 2 Extreme points of the polyhedron

Require: Matrix $A \in \mathbb{R}_+^{m \times n}$, vector $b \in \mathbb{R}_+^m$
Ensure: Set of extreme points $\mathcal{V}$ of polytope $\mathcal{P} = \{x \mid Ax^T \leq b\}$
1: Initialize $\mathcal{V} \leftarrow \emptyset$
2: **for all** subsets $\mathcal{I} \subseteq \{1, \ldots, m\}$ such that $|\mathcal{I}| = n$ **do**
3: Construct submatrix $A' = A_{\mathcal{I}}$ and vector $b' = b_{\mathcal{I}}$ ▷ Select a subset of n inequalities
4: Consider the linear system
$$A'x^T = b'$$
5: **if** $\left\{x \in \mathbb{R}^n \ : \ A'x^T = b'\right\} = \emptyset$ **then**
6: **goto** step 2. ▷ Case (a): The system has no solution
7: **end if**
8: **if** $\text{rank}(A') < n$ **then**
9: **goto** step 2. ▷ Case (b): The system does not have a unique solution
10: **end if**
11: Let $x^{\mathcal{I}}$ be the unique solution of $A'x^T = b'$
12: **if** $Ax^{\mathcal{I}} \leq b$ **then**
13: Add $x^{\mathcal{I}}$ to $\mathcal{V}$ ▷ Case (c): unique solution exists
14: **end if**
15: **end for**
16: **return** $\mathcal{V}$

Consider the example presented in Figure 3.4, where a two–dimensional polyhedron is presented, defined by four inequalities of the form $a_i\boldsymbol{x} \leq b_i$, along with hyperplanes $x_1 \geq 0$ and $x_2 \geq 0$ as follows:

$$\begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ a_{13} & a_{23} \\ a_{14} & a_{24} \\ -1 & 0 \\ 0 & -1 \end{bmatrix} [x_1 \ x_2]^T \leq \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ 0 \\ 0 \end{bmatrix}$$

As depicted in the figure, the polyhedron $\mathcal{P}$ possesses six vertices marked in green. Vertices marked in red correspond to intersections of hyperplanes that satisfy the system of equations but do not belong to the feasible region of the polyhedron.

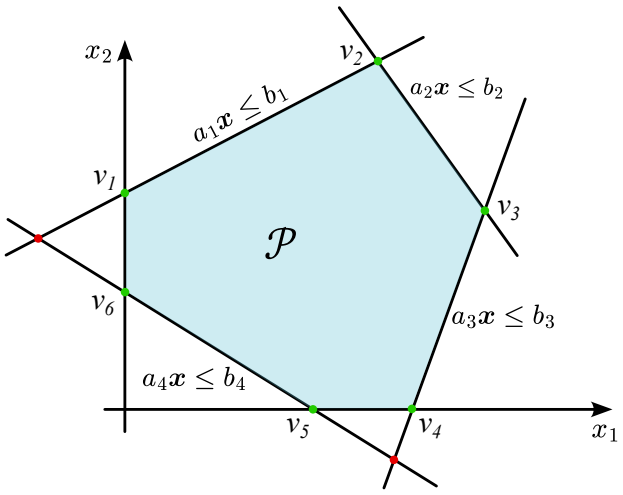


Figure 3.4: Illustration of a continuous polyhedron in $\mathbb{R}^2$ defined by six hyperplanes ($m = 6$).

For instance, the vertex v_2 is obtained by solving the system of linear inequalities composed by $a_1\boldsymbol{x} \leq b_1$ and $a_2\boldsymbol{x} \leq b_2$ yielding a unique solution. Note that, for this particular example, cases (a) and (b) of Algorithm 2 do not apply, since every hyperplane intersects at some point in $\mathbb{R}^2$, and none of the inequalities are linearly dependent.
The specific procedure employed to determine the vertices of the polyhedron $\mathcal{P}$ defined in (3.1) is presented in Algorithm 3. The main difficulty associated with this formulation lies in the combinatorial nature of the underlying feasible set, which arises from the interaction between binary variables and continuous decision variables. In this way and in general terms, the algorithm can be decomposed into three main stages. First, candidate vertices are generated by solving for the continuous variables under the activation of a single GUB constraint. Second, additional candidate vertices are obtained by determining the values of the continuous variables when two GUB constraints are simultaneously active. Finally, a feasibility filtering step is performed to retain only those vectors that satisfy all constraints of the original polyhedron, thereby ensuring that the resulting set consists exclusively of feasible vertices of $\mathcal{P}$.
The previous section presented the procedure used to determine the extreme points of the polyhedron. In the next section, we describe the second step, which consists of determining the hyperplane representation of the polyhedron using specialized software tools.

3.3.2 Hyperplane representation

In the present section we describe the methodology applied to perform the polyhedral analysis of the polyhedron $\mathcal{P}$, based on the software PORTA. As previously noted, PORTA takes as input a finite set of points in any given dimension and computes the corresponding convex–hull.
In this way, the polyhedron $\mathcal{P}$ becomes extremely difficult to characterize as the dimension increases, due to the rapid growth in the number of vertices and the resulting combinatorial complexity. This motivates the use of computational tools such as the aforementioned software, which are particularly useful for gaining insight into the underlying polyhedral structure.
The proposed framework involves generating random instances of fixed dimensionality and analyzing the resulting convex hull of each instance, as computed by the software PORTA. By considering a sufficiently large pool of instances, it becomes possible to identify recurring patterns and discern similarities among the inequalities, which may correspond to a specific templates of the convex–hull for the given dimension. Once a candidate template is identified and the corresponding parameters are determined, it is mandatory to rigorously determine the conditions under which this template defines a facet of the convex–hull.

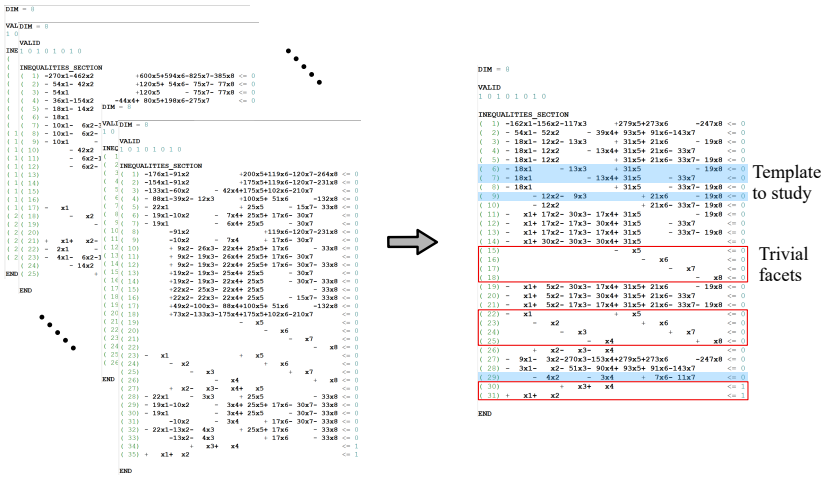


Figure 3.5: PORTA analysis used to characterize the polyhedron, where the blue shaded inequalities correspond to a particular template to analyze, while the ones boxed in red correspond to the trivial facets of the polyhedron.

Algorithm 3 Enumeration and filtering of candidate vertices

Require: Vector $a \in \mathbb{R}^n$, vector $m \in \mathbb{R}^n$, scalar δ, dimension n
Ensure: Set of feasible vertices $\mathcal{V}$

Initialize $\mathcal{S} \leftarrow \{\mathbf{0}_{2n}\}$
Generate all binary vectors $\mathcal{X} \leftarrow \{x \in \{0,1\}^n\}$
Define $y \leftarrow$ symbolic
for all $x \in \mathcal{X}$ **do**
    if $\sum_i x_i = 1$ **then**
        Let $e \leftarrow \{i : x_i = 1\}$
        $y \leftarrow \text{linsolve}(a^\top x + m_e y = \delta,\ y)$
        Construct y^* with $y^*_e = y$, $y^*_k = 0$ for $k \neq e$
        $\mathcal{S} \leftarrow \mathcal{S} \cup \{(x, y^*)\}$
        for all product$\{(b_1, b_2) : b_1 \in \{0,1\},\ b_2 \in \{0,1\}\}$ **do**
            $y \leftarrow b_1$
            Construct y^* with $y^*_e = b$, $y^*_k = 0$ for $k \neq e$
            $\mathcal{S} \leftarrow \mathcal{S} \cup \{(x, y^*)\}$
        end for
    else
        Let $e = \{i, j\}$ be the indices such that $x_i = x_j = 1$
        for all $(i, j) \in e \times e,\ i \neq j$ **do**
            for all $\bar{y} \in \{0,1\}$ **do**
                $y \leftarrow \text{linsolve}(a^\top x + m_j \bar{y} + m_i y = \delta,\ y)$
                Construct y^* with $y^*_i = y$, $y^*_j = \bar{y}$
                $\mathcal{S} \leftarrow \mathcal{S} \cup \{(x, y^*)\}$
            end for
        end for
        for all product$\{(b_1, b_2) : b_1 \in \{0,1\},\ b_2 \in \{0,1\}\}$ **do**
            $y^* \leftarrow y^*_i = b_1$, $y^*_j = b_2$, $y^*_k = 0$ for $k \neq e$
            $\mathcal{S} \leftarrow \mathcal{S} \cup \{(x, y^*)\}$
        end for
    end if
end for
Initialize $\mathcal{V} \leftarrow \emptyset$
for all $(x, y) \in \mathcal{S}$ **do** ▷ Feasibility filtering
    if $a^\top x + m^\top y \leq \delta$ **and** $0 \leq y \leq 1$ **then**
        $\mathcal{V} \leftarrow \mathcal{V} \cup \{(x, y)\}$
    end if
end for
return $\mathcal{V}$

Figure 3.5 illustrates the methodology associated with the PORTA–based polyhedral analysis. The right–hand side of the figure represents the database of instances under study, whereas the left–hand side corresponds to a particular instance characterized by the parameter vectors a equal to $[7, 13, 17, 17]$, a matrix b equal to $[31, 21, 33, 19]$, and a gradient δ equal to 8. In this representation, the inequalities highlighted by red boxes correspond to trivial facets, while those enclosed by shaded rectangles identify a specific inequality template selected for further analysis.
Trivial facets correspond to those inequalities of the polyhedron that are already explicitly included in its defining formulation, as given in (3.1). While they define supporting hyperplanes of the polyhedron, they do not provide additional insight into its underlying polyhedral structure beyond what is already encoded in the formulation. Furthermore, the procedure for analyzing a given inequality template is based on identifying groups of inequalities that exhibit a common pattern in the indexing of the variables, and subsequently determining the manner in which the associated parameters are constructed. For instance, the template analyzed in Figure 3.5 exhibits a structured pattern in the variable indices. In particular, the GUB inequality associated with the edges entering the node is simultaneously active for both the integer and the continuous variables. In contrast, for the edges leaving the node, the indices of the continuous variables correspond to the complement of the indices of the associated binary variables. Subsequently, the conditions under the given inequality defines a facet must be established in order to properly characterize the polyhedron,
As a summary of the methodology applied to derive the facets of the polyhedron, Figure 3.6 presents a schematic overview of the relevant steps described above. In particular, steps 1, 2, and 3 are discussed in the present section, while step 4 corresponds to the mathematical proofs, which are presented and discussed in the subsequent section.

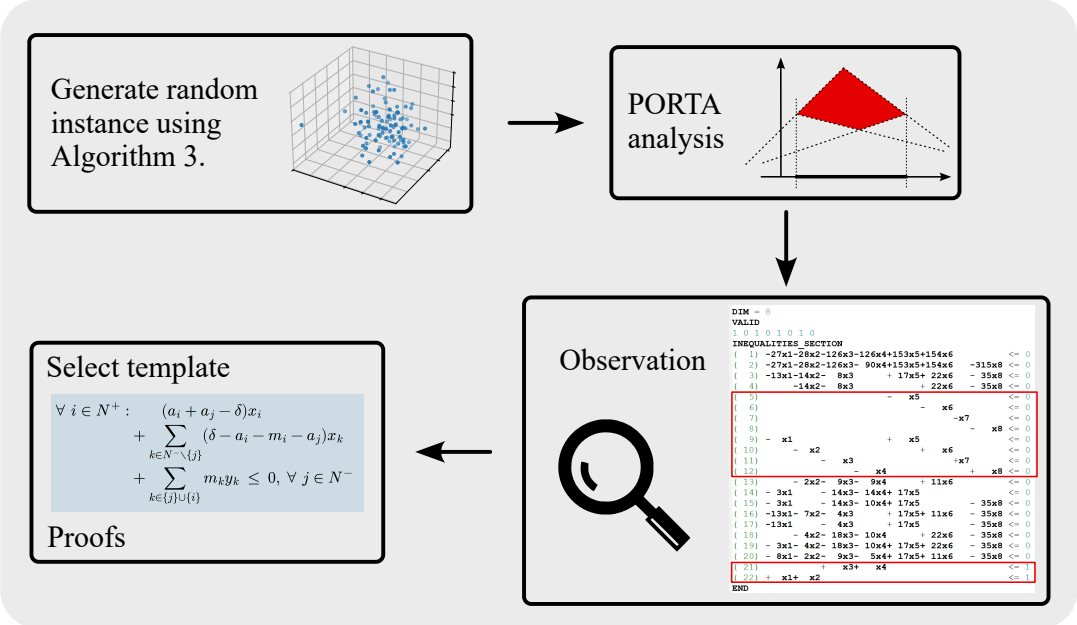


Figure 3.6: Schematic summary of the four principal steps of the methodology adopted to derive the facet–defining inequalities of the polyhedron under study, based on the scientifi method.

3.4 Polyhedral results

In this section we introduce some assumptions and we describe the trivial facet–defining inequalities with the respective proof and the cases where applies if necessary. For our propositions, we will

define two relevant cases: *i*) $\mathcal{M} = \{k \in N \,:\, a_k,\, m_k < 0\}$, and *ii*) $\mathcal{M} = \{k \in N \,:\, a_k,\, m_k > 0\}$, which can be interpreted as switching the flow within the node. In this context, to present a more compact notation of the subsequent results, let us define N^+ as the subset of N that contains positive parameters, and conversely, N^- as the subset of N that contains negative parameters, were $N^+,\, N^- \subset N$ and $N^+ \cap N^- = \emptyset$.
Additionally, for the upcoming proofs and without loss of generality, the semi–continuous knapsack constraint in (3.1) can be reformulated as: $\sum_{k\in N^+}(a_k x_k + m_k y_k) - \sum_{k\in N^-}(a_k x_k + m_k y_k) \leq \Delta$, where $a_k,\, m_k$ and Δ are in $\mathbb{R}^+$. The following assumptions are considered:

Assumption 1 $n \geq 2$;

Assumption 2 $a_k \neq\ 0\ \forall\, k \in N$;

Assumption 3 $\text{sign}\{a_k\} = \text{sign}\{m_k\}\ \forall k \in N$;

Assumption 4 $\forall\, i \in N^+,\, \exists\, j \in N^- : a_i + a_j < \Delta - m_j$.

When Assumption 1 is not satisfied, either the set $\mathcal{M}$ or $N \setminus \mathcal{M}$ is empty. This contradicts the definition of the set $\mathcal{M}$, so we can assume without loss of generality that Assumption 1 holds. If $a_j = 0$ for some $j \in N$, x_j can be eliminated from the problem. Therefore, there is no loss of generality in Assumption 2. On the same argument used for Assumption 2, there is no loss of generality in Assumption 4. In particular, note that $a_k,\ \forall k \in N^+$, is the lower bound of the semi–continuous variable entering to the node, as well, $a_k + m_k,\ \forall N^-$, is the upper bound of the semi–continuous variable leaving the node. If $\exists\, k \in N^+$ such that $a_k - (a_j + m_j) > \Delta,\ \forall j \in N^-$, the variables x_k and y_k will never be active and can be eliminated from the problem, along with y_k. With this in place, Proposition 2 through Proposition 6 can be derived, as detailed in the following section, where the corresponding proofs are provided for each of the result.

3.4.1 Trivial proofs

Proposition 2 *The polyhedron $\mathcal{P}$ is full–dimensional.*

Proof To prove that the set $\mathcal{P}$ is full dimensional, lets define ϵ such that $0 < \epsilon \leq 1$. Then, $\forall k \in N^-$, construct $\boldsymbol{p_k} = \{(\boldsymbol{e_k}, \boldsymbol{0_n}), (\boldsymbol{e_k}, \boldsymbol{e_k})\}$. Additionally, $\forall k \in N^+$ define $\boldsymbol{q^k} = (\boldsymbol{e_k} + \boldsymbol{e_{k0}}, \boldsymbol{e_{k0}})$, and, $\boldsymbol{r^k} = (\boldsymbol{e_k} + \boldsymbol{e_{k0}},\, \epsilon\boldsymbol{e_k} + \boldsymbol{e_{k0}})$, where $\boldsymbol{e_{k0}} = \text{argmin}\{a_k \,:\, k \in N^-\}$, i.e. the index k in the set N^- such that a_k is the smallest among all a_k with $k \in N^-$. Also, let be $\boldsymbol{s} = (\boldsymbol{0_n}, \boldsymbol{0_n})$. The points $\boldsymbol{p^k}$, $\boldsymbol{q^k}$ and, $\boldsymbol{r^k}$, and $\boldsymbol{s}$ belong to $\mathcal{P}$ because of equation (3.1). These points are affinely independent since $\forall k \in N$, $\boldsymbol{p^k} - \boldsymbol{s}$ ($2|N^-|$ points), $\boldsymbol{q^k} - \boldsymbol{s}$ ($|N^+|$ points), and $\boldsymbol{r^k} - \boldsymbol{s}$ ($|N^+|$ points) are linearly independent point in $\mathcal{P}$. Since we have described $2n + 1$ affinely independent points in $\mathcal{P}$, we have shown that the set $\mathcal{P}$ is full–dimensional.

■

Proposition 3 *The inequality $y_k \geq 0$ is facet-defining for $\mathcal{P}$, $\forall k \in N$.*

Proof To demonstrate that the inequality $y_k \geq 0$ defines facets of the polyhedron $\mathcal{P}$, we will consider the two different cases: *i*) the case where $k \in N^+$, and *ii*) the case where $k \in N^-$. Also, let $\bar{j}$ be an index in N^- such that $\bar{j} = \text{argmin}\{a_k,\, k \in N^-\}$

i) For the first scenario, and for any given index $k \in N^+$ consider the following set of affinely independent points the set $\mathcal{P}$:

$$\boldsymbol{q}_j = \begin{cases} (\boldsymbol{e}_j,\, \boldsymbol{0}_n) \\ (\boldsymbol{e}_j,\, \boldsymbol{e}_j) \end{cases} \quad ; \quad \forall i \in N^-$$

$$\boldsymbol{p}_{i,j} = \begin{cases} (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \boldsymbol{e}_j) \\ (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \epsilon\boldsymbol{e}_i + \boldsymbol{e}_{\bar{j}}) \end{cases} ; \quad \begin{array}{l} \forall i \in N^+ \setminus \{k\} \\ j = \bar{j} \end{array}$$

$$\boldsymbol{r}_{i,j} = \; (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \boldsymbol{e}_j) \quad ; \quad \begin{array}{l} i = k \\ j = \bar{j} \end{array}$$

ii) For the second scenario, and for any given index $k \in N^-$ consider the following set of affinely independent points in $\mathcal{P}$:

$$\boldsymbol{q}_j = \begin{cases} (\boldsymbol{e}_j,\, \boldsymbol{0}_n), & \forall j \in N^- \\ (\boldsymbol{e}_j,\, \boldsymbol{e}_j), & \forall j \in N^- \setminus \{k\} \end{cases}$$

$$\boldsymbol{p}_{i,j} = \begin{cases} (\boldsymbol{e}_j + \boldsymbol{e}_j,\, \boldsymbol{e}_j) \\ (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \epsilon\boldsymbol{e}_j + \boldsymbol{e}_j) \end{cases} ; \quad \begin{array}{l} \forall i \in N^+, \\ \forall j \in N^- \setminus \{k\} \end{array}$$

$$\boldsymbol{r}_{i,j} = \; (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \epsilon\boldsymbol{e}_i) \quad ; \quad \begin{array}{l} \forall i \in N^+ \\ j = k \end{array}$$

For both cases, we additionally consider the point $s_n = \boldsymbol{0}_n$. Let $\epsilon \in (0, 1]$ be a scalar parameter such that, for any admissible value of ϵ , the vector under consideration belongs to the polyhedron $\mathcal{P}$. Note that the specific value of ϵ associated with each point need not be identical across all points. Considering that the points $\boldsymbol{q}_j$, $\boldsymbol{p}_{i,\bar{j}}$, $\boldsymbol{r}_{i,j}$ and $\boldsymbol{s}_n$, are affinely independent and that their cardinality equals the dimension of the space, the inequality $y_k \geq 0$, $\forall k \in N^+ \cup N^-$, define a facet of the polyhedron.

■

Proposition 4 *To demonstrate that the inequality $x_k \geq y_k$ defines facets of the polyhedron $\mathcal{P}$, we proceed analogously to the previous case by distinguishing between two different cases: i) the case where $k \in N^+$, and ii) the case where $k \in N^-$.*

i) For the first scenario, the inequality $x_k \geq y_k$, $\forall k \in N^+$ defines a facet of the polyhedron if $\exists\, j \in N^-$ such that: $a_k + m_k + a_j + m_j \leq \Delta$. When this condition is satisfied, consider the following set of affinely independent points of $\mathcal{P}$:

$$\boldsymbol{q}_j = \begin{cases} (\boldsymbol{e}_j,\, \boldsymbol{0}_n) \\ (\boldsymbol{e}_j,\, \boldsymbol{e}_j) \end{cases} ; \quad \forall j \in N^-$$

$$\boldsymbol{p}_{i,j} = \begin{cases} (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \boldsymbol{e}_j) \\ (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \epsilon\boldsymbol{e}_i + \boldsymbol{e}_j) \end{cases} ; \quad \begin{array}{l} \forall i \in N^+ \setminus \{k\} \\ j = \bar{j} \end{array}$$

$$\boldsymbol{r}_{i,j} = \; (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \boldsymbol{e}_i + \boldsymbol{e}_j) \quad ; \quad \begin{array}{l} i = k, \\ j = \bar{j} \end{array}$$

ii) For the second scenario, and for any given index $k \in N^-$ consider the following set of affinely independent points in $\mathcal{P}$:

$$\boldsymbol{q}_j = \begin{cases} (\boldsymbol{e}_j,\, \boldsymbol{0}_n), & \forall j \in N^- \\ (\boldsymbol{e}_j,\, \boldsymbol{e}_j), & \forall j \in N^- \setminus \{k\} \end{cases}$$

$$\boldsymbol{p}_{i,j} = \begin{cases} (\boldsymbol{e}_j + \boldsymbol{e}_j,\, \boldsymbol{e}_j) \\ (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \epsilon\boldsymbol{e}_i + \boldsymbol{e}_j) \end{cases} ; \quad \begin{array}{l} j = k \\ \forall i \in N^+ \end{array}$$

For both cases, we additionally consider the point $s_n = \boldsymbol{0}_n$. Let $\epsilon \in (0, 1]$ be a scalar parameter such that, for any admissible value of ϵ , the vector under consideration belongs to the polyhedron $\mathcal{P}$. Note that the specific value of ϵ associated with each point need not be identical across all points. Considering that the points $\boldsymbol{q}_j$, $\boldsymbol{p}_{i,j}$, $\boldsymbol{r}_{i,j}$ and $\boldsymbol{s}_n$, are affinely independent and that their cardinality equals the dimension of the space, the inequality $x_k \geq y_k$, $\forall k \in N^+ \cup N^-$, define a facet of the polyhedron, under the given conditions.

■

Proposition 5 *The inequality* $\sum_{N^-} x_k \leq 1$ *is facet-defining for* $\mathcal{P}$*. Consider the following set of affinely independent points in the polyhedron:*

$$\boldsymbol{q}_j = \begin{cases} (\boldsymbol{e}_j,\, \mathbf{0}_n) \\ (\boldsymbol{e}_j,\, \boldsymbol{e}_j) \end{cases} \quad ; \quad \forall j \in N^-$$

$$\boldsymbol{p}_{i,j} = \begin{cases} (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \boldsymbol{e}_j) \\ (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \epsilon \boldsymbol{e}_i + \boldsymbol{e}_j) \end{cases} \quad ; \quad \begin{array}{l} \forall i \in N^+ \\ j \in N^- \end{array}$$

Considering that the points $\boldsymbol{q}_j$ *and* $\boldsymbol{p}_{i,j}$*, are affinely independent and that their cardinality equals the dimension of the space, the GUB inequality* $\sum_{N^-} x_k \leq 1$ *is facet–defining for the polyhedron. Also let* $\epsilon \in (0, 1]$ *be a scalar parameter such that, for any admissible value of* ϵ *, the vector under consideration belongs to the polyhedron* $\mathcal{P}$*. Note that the specific value of* ϵ *associated with each point need not be identical across all points.*

■

Proposition 6 *If* $\Delta \geq \min\{a_k : k \in N^+\}$*, the GUB inequality given by* $\sum_{N^+} x_k \leq 1$ *defines facet of the* $\mathcal{P}$*. Also, let* $\bar{i}$ *be an index in* N^+ *such that* $\bar{i} = \text{argmin}\{a_k,\, k \in N^+\}$*.*

$$\boldsymbol{q}_i = \; (\boldsymbol{e}_i,\, \mathbf{0}_n) \quad ; \quad i \in N^+$$

$$\boldsymbol{p}_{i,j} = \begin{cases} (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \boldsymbol{e}_j) \\ (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \epsilon \boldsymbol{e}_j) \end{cases} \quad ; \quad \begin{array}{l} \forall i \in N^+ \\ \forall j \in N^- \end{array}$$

$$\boldsymbol{r}_{i,j} = \; (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \epsilon \boldsymbol{e}_i + \boldsymbol{e}_j) \quad ; \quad \begin{array}{l} i \in N^+ \\ j = \bar{j} \end{array}$$

Considering that the points $\boldsymbol{q}_j$*,* $\boldsymbol{p}_{i,j}$ *and* $\boldsymbol{r}_{i,j}$*, are affinely independent and that their cardinality equals the dimension of the space, the GUB inequality* $\sum_{N^+} x_k \leq 1$ *is facet–defining for the polyhedron under under the given conditions. Also let* $\epsilon \in (0, 1]$ *be a scalar parameter such that, for any admissible value of* ϵ *, the vector under consideration belongs to the polyhedron* $\mathcal{P}$*. Note that the specific value of* ϵ *associated with each point need not be identical across all points.*

■

3.4.2 Cutting planes formulation

For the proofs that follow, we adopt a simplified notational convention. Variables $\boldsymbol{x}$ and $\boldsymbol{y}$ are written with a single subscript, even when they are implicitly associated with an entering index i or a leaving index j. In particular, when these variables appear in summations over N^+ or N^- , the second subscript identifying the corresponding set is omitted for the sake of notational simplicity, as it is clear from the summation context whether the index belongs to N^+ or $n \in N^-$.

Proposition 7 *For any given node* n *in a graph* $\mathcal{G}$*, let* (i', j') *be an arbitrary element of* $N_n^+ \times N_n^-$ *respecting the GUB inequalities and the stated assumptions. The following inequality is facet-defining for the associated polyhedron provided that the following conditions are simultaneously satisfied: i) for* $i = i'$ *and* $j = j'$*:* $\exists\, \epsilon \in (0, 1]$ *such that:* $a_i + \epsilon m_i + a_j < \delta$*, and ii) for* $i \in N_n^+ \setminus \{i'\}$ *and for* $j = j'$*:* $0 < \frac{\delta - a_i - m_i - a_j}{m_j} < 1 \wedge 0 < \frac{\delta - a_i - a_j}{m_i} < 1$*.*

$$\begin{aligned} & \sum_{k \in N_n^+ \setminus \{i'\}} (a_k + a_{j'} - \delta) x_k \\ + & \sum_{k \in N_n^- \setminus \{j'\}} \min\left\{\delta - a_\nu - m_\nu - a_{j'},\, \nu \in N_n^+\right\} x_k \\ + & \sum_{k \in N_n^+ \setminus \{i'\} \cup \{j'\}} m_k y_k \;\leq\; 0 \end{aligned} \tag{3.2}$$

Proof *To show that the above inequality is facet–defining for the polyhedron $\mathcal{P}$, consider the fixed pair let $(i', j') \in N_n^+ \times N_n^-$. For this pair, we consider the following set of affinely independent points:*

$$\boldsymbol{q}_{i,j} = \begin{cases} (\boldsymbol{e}_i + \boldsymbol{e}_j,\ \mathbf{0}_n) & ;\ j = j' \\ (\boldsymbol{e}_i + \boldsymbol{e}_j,\ \epsilon \boldsymbol{e}_i) & ;\ i = i' \end{cases}$$

$$\boldsymbol{r}_{i,j} = \begin{cases} (\boldsymbol{e}_i + \boldsymbol{e}_j,\ \epsilon \boldsymbol{e}_i + \boldsymbol{e}_j) & ;\ \forall\, j \in N_n^- \setminus \{j'\} \\ (\boldsymbol{e}_i + \boldsymbol{e}_j,\ \epsilon_1 \boldsymbol{e}_i + \epsilon_2 \boldsymbol{e}_j) & ;\ i = \operatorname{argmin}\{a_k,\ k \in N_n^+ \setminus \{i'\}\} \end{cases}$$

$$\boldsymbol{o}_{i,j} = \begin{cases} (\boldsymbol{e}_i + \boldsymbol{e}_j,\ \epsilon \boldsymbol{e}_i) & ;\ j = j' \\ (\boldsymbol{e}_i + \boldsymbol{e}_j,\ \boldsymbol{e}_i + \epsilon \boldsymbol{e}_j) & ;\ i \in N_n^+ \setminus \{i'\} \end{cases}$$

Consider also the point $\boldsymbol{s}_n$, as defined before and the point $p_j = (\boldsymbol{e}_j,\ \mathbf{0}_n)$ defined for $j = j'$. Let also $\epsilon \in (0, 1]$ be a scalar parameter such that, for any admissible value of ϵ, the vector under consideration belongs to the polyhedron $\mathcal{P}$, and moreover it is associated to a tight vector with respect to the given pair (i', j') for the inequality in (3.2)*. Note that the specific value of ϵ associated with each point need not be identical across all points.*

■

Example 1 *Let $conv(\mathcal{P})$ be the convex hull of points satisfying*

$$\begin{aligned} \mathcal{P} = \{(x, y) \in \{0,1\}^n \times [0,1]^n :\ & 7x_1 + 14y_1 + 7x_2 + 12y_2 - 12x_3 - 18y_3 - 12x_4 - 13y_4 \le 5, \\ & \textstyle\sum_{k \in N^+} x_k \le 1, \\ & \textstyle\sum_{k \in N^-} x_k \le 1, \\ & y_k \le x_k,\ \forall\, k \in N\} \end{aligned} \tag{3.3}$$

where $N^+ = \{1, 2\}$ and $N^- = \{3, 4\}$. Consider the following system of linear inequalities, which characterizes the convex hull $conv(\mathcal{P})$ of the polyhedron, excluding its trivial facets:

$$-10x_1 - 10x_2 \qquad\qquad +14y_1 + 12y_2 - 18y_3 - 13y_4 \le 0 \tag{3.4}$$
$$-10x_1 \qquad\qquad +14y_1 \qquad - 18y_3 - 13y_4 \le 0 \tag{3.5}$$
$$-10x_2 \qquad\qquad + 12y_2 - 18y_3 - 13y_4 \le 0 \tag{3.6}$$
$$+x_1 + x_2 \quad -x_3 - x_4 \qquad\qquad\qquad\qquad \le 0 \tag{3.7}$$
$$-10x_2 - 2x_3 \qquad\qquad + 12y_2 \qquad - 13y_4 \le 0 \tag{3.8}$$
$$-10x_1 - 10x_2 - 4x_3 \qquad +14y_1 + 12y_2 \qquad - 13y_4 \le 0 \tag{3.9}$$
$$-10x_1 \qquad -4x_3 \qquad +14y_1 \qquad\qquad - 13y_4 \le 0 \tag{3.10}$$
$$\mathbf{-5x_2} \qquad \mathbf{-x_4} \qquad \mathbf{+6y_2\ -9y_3} \qquad\qquad \mathbf{\le 0} \tag{3.11}$$
$$-5x_1 - 10x_2 - 2x_3 \qquad +7y_1 + 12y_2 \qquad - 13y_4 \le 0 \tag{3.12}$$
$$-5x_1 - 10x_2 \qquad - 2x_4 \ +7y_1 + 12y_2 - 18y_3 \qquad\qquad \le 0 \tag{3.13}$$
$$-5x_1 \qquad\qquad - 2x_4 \ +7y_1 \qquad - 9y_3 \qquad\qquad \le 0 \tag{3.14}$$
$$-5x_1 - 5x_2 \qquad - 2x_4 \ +7y_1 + 6y_2 \quad - 9y_3 \qquad\qquad \le 0 \tag{3.15}$$

First, observe that for any pair $(i', j') \in N_n^+ \times N_n^-$ all standing assumptions are satisfied in the present case, as well as the specific conditions required for the inequality to be facet-defining. And in particular the facets corresponding to (3.2) *are* (3.14)*,* (3.11)*,* (3.8) *and* (3.10)*.*
Let us consider facet (3.11)*, highlighted in bold, to demonstrate that there exists a set of tight points satisfying the inequality as an equality, with cardinality equal to the dimension of the space and also are contained in the polyhedron. The following vectors in the set $\mathcal{S}$ are feasible points that satisfy inequality* (3.2) *with equality.*

$$\begin{aligned} \mathcal{S} = \ & \{(0,0,0,0,0,0,0,0), (0,0,0,1,0,0,0,0), (0,1,0,1,0,1,0,1), (0,1,0,1,0,1,0,0.99), \\ & (0,1,1,0,0,0.83,0,0), (0,1,1,0,0,0.9,0.04,0), (1,0,0,1,0,0,0,0), (1,0,0,1,0.2,0,0,0)\} \end{aligned}$$

given that $|\mathcal{S}| = dim(\mathcal{P})$, the linear constraint (3.11) *corresponds to a facet of the convex hull of $\mathcal{P}$. The same strategy can be used to prove that the inequalities* (3.11), (3.8), *and* (3.10) *each define a facet of the polyhedron under study.*

Proposition 8 *For any given node n in a graph $\mathcal{G}$, let $j' \in N_n^-$. The following inequality is facet-defining for the associated polyhedron provided that the following conditions are simultaneously satisfied: i) for $j = j'$ and for any $i \in N_n^+$:$a_i + m_i + a_{j'} + \epsilon m_{j'} < \delta$, and ii) for $j \in N_n^- \setminus \{j'\}$, $\exists\, i \in N_n^+$ such that $a_i + a_j - \delta + \min\{\delta - a_k - m_k - a_j,\, k \in N_n^+\} < m_i$.*

$$\begin{aligned} & \sum_{k \in N_n^+} (a_k + a_{j'} - \delta)\, x_k \\ + & \sum_{k \in N_n^- \setminus \{j'\}} \min\{\delta - a_\nu - m_\nu - a_{j'},\, \nu \in N_n^+\} x_k \\ + & \sum_{k \in N_n^+ \cup \{j'\}} m_k y_k \;\leq\; 0 \end{aligned} \tag{3.16}$$

Proof *To construct a set of affinely independent points in $\mathcal{P}$ that are tight for the given inequality, let $i^* \in N_n^+$ denote an index satisfying condition (i) stated in the proposition. Under this assumption, we consider the following set of affinely independent points:*

$$\boldsymbol{q}_{i,j} = \begin{cases} (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \epsilon \boldsymbol{e}_i + \boldsymbol{e}_j) & ; \quad \forall\, j \in N_n^- \setminus \{j'\} \\ (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \epsilon_1 \boldsymbol{e}_i + \epsilon_2 \boldsymbol{e}_j) & ; \quad i = i^* \end{cases}$$

$$\boldsymbol{r}_{i,j} = \begin{cases} (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \epsilon \boldsymbol{e}_i) & ; \quad j = j' \\ (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \boldsymbol{e}_i + \epsilon \boldsymbol{e}_j) & ; \quad i \in N_n^+ \end{cases}$$

Consider also the point $\boldsymbol{s}_n$, as defined before and the point $p_j = (\boldsymbol{e}_j,\, \boldsymbol{0}_n)$ defined for $j = j'$. Let $\epsilon \in (0, 1]$ be a scalar parameter such that, for any admissible value of ϵ, the vector under consideration belongs to the polyhedron $\mathcal{P}$, and moreover it is associated to a tight vector with respect to the given index j' for the inequality in (3.16).

■

Proposition 9 *For any given node n in a graph $\mathcal{G}$, let $i' \in N_n^+$. The following inequality is facet-defining for the associated polyhedron provided that the following conditions are simultaneously satisfied: i) Let $i' \in N_n^+$, for every $j \in N_n^-$, there exists a parameter $\alpha \in [0, 1]$ such that: $a_{i'} + m_{i'}\alpha + a_j + m_j \leq \delta$; ii) for any pair $(i', j') \in N_n^+ \times N_n^-$, there exists β such that: $a_{i'} + m_{i'} + a_{j'} + \beta m_{j'} \geq \delta$; and iii) for a given pair $(i', j') \in N_n^+ \times N_n^-$, the following condition holds for all $j \in N_e^- \setminus \{j'\}$*

$$\begin{aligned} & (a_{i'} + a_{j'} - \delta) x_n \\ + & \sum_{k \in N^- \setminus \{j'\}} (\delta - a_{i'} - m_{i'} - a_{j'}) x_k \\ + & \sum_{k \in \{i'\} \cup \{j'\}} m_k y_k \;\leq\; 0, \qquad \forall\, j' \in N^-. \end{aligned} \tag{3.17}$$

Proof *To prove that the valid inequality defined in* (3.17) *defines facets of the polyhedron $\mathcal{P}$ let's define a number equal to dim(N) affinely independent point who are feasible for the polyhedron. Let consider following set of points:*

$$\boldsymbol{q}_{i,j} = \begin{cases} (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \epsilon \boldsymbol{e}_i) & ; \quad i = i' \\ (\boldsymbol{e_i} + \boldsymbol{e_j},\, \epsilon_1 \boldsymbol{e}_i + \epsilon_2 \boldsymbol{e}_j) & ; \quad j = j' \end{cases}$$

$$\boldsymbol{r}_{i,j} = \begin{cases} (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \epsilon\boldsymbol{e}_i + \boldsymbol{e}_j) \\ (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \epsilon_1\boldsymbol{e}_i + \epsilon_2\boldsymbol{e}_j) \end{cases} ; \quad \begin{array}{l} i = i' \\ j \in N_n^- \setminus \{j'\} \end{array}$$

$$\boldsymbol{o}_{i,j} = \begin{cases} (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \mathbf{0}_n) \\ (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \epsilon\boldsymbol{e}_i) \end{cases} ; \quad \begin{array}{l} i \in N_n^+ \setminus \{i'\} \\ j = j' \end{array}$$

Consider also the point $\boldsymbol{s}_n$, as defined before and the point $p_j = (\boldsymbol{e}_j,\, \mathbf{0}_n)$ defined for $j = j'$. Let $\epsilon \in (0, 1]$ be a scalar parameter such that, for any admissible value of ϵ, the vector under consideration belongs to the polyhedron $\mathcal{P}$, and moreover it is associated to a tight vector with respect to the given index j' for the inequality in (3.17)*. Note that the specific value of ϵ associated with each point need not be identical across all points.*

■

Proposition 10 *For any given node n in a graph $\mathcal{G}$, let $\bar{j} = \text{argmin}\{a_k,\, k \in N^-\}$. The following inequality is facet–defining for the associated polyhedron provided that the following conditions are simultaneously satisfied: i) $a_i + m_i + a_{\bar{j}} + m_{\bar{j}} \leq \delta,\ \forall\, i \in N_n^+$, and ii) $\exists\, i \in N_n^+$, such that: $a_i + m_i + a_j + m_j \geq \delta,\ \forall\, j \in N_n^- \setminus \{\bar{j}\}$.*

$$\sum_{k \in N_n^+} \left(a_k + a_{\bar{j}} - \delta\right) x_k + \sum_{k \in N_n^+ \cup \{\bar{j}\}} m_k y_k + \sum_{k \in N_n^- \setminus \{\bar{j}\}} \left(a_k + m_k - a_{\bar{j}}\right) y_k \;\leq\; 0 \tag{3.18}$$

Proof *To prove that inequality in* (3.18) *defines a facet of the polyhedron $\mathcal{P}$, under the given conditions, let's define a number equal to dim(N) affinely independent point who are feasible for the polyhedron. Let consider following set of points:*

$$\boldsymbol{q}_{i,j} = \begin{cases} (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \epsilon\boldsymbol{e}_i) \\ (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \epsilon_1\boldsymbol{e}_i + \epsilon_2\boldsymbol{e}_j) \end{cases} ; \quad \begin{array}{l} i \in N_n^+ \\ j = \bar{j} \end{array}$$

$$\boldsymbol{r}_{i,j} = \;\; (\boldsymbol{e}_i + \boldsymbol{e}_j,\, \epsilon\boldsymbol{e}_i + \boldsymbol{e}_i) \quad ; \quad \begin{array}{l} \exists\, i \in N_n^+ \\ j \in N_n^- \setminus \{\bar{j}\} \end{array}$$

Consider also the point $\boldsymbol{s}_n$, as defined before and the point $p_j = (\boldsymbol{e}_j,\, \mathbf{0}_n)$ defined for $\forall j \in N_n^-$. Let $\epsilon \in (0, 1]$ be a scalar parameter such that, for any admissible value of ϵ, the vector under consideration belongs to the polyhedron $\mathcal{P}$, and moreover it is associated to a tight vector with respect to the given index j' for the inequality in (3.18)*. Note that the specific value of ϵ associated with each point need not be identical across all points.*

■

Chapter 4

Computational experiments

This chapter presents the computational experiments conducted to evaluate the performance of the different formulations under consideration, namely the standard NF formulation, the *3bin* formulation, and the enhanced formulation proposed in this work. First, an overall description of the formulations is provided, together with the performance metrics used to assess their relative quality. These metrics include measures related to relaxation strength, integrality gap, and computational efficiency. The computational setup under which the experiments were performed is also detailed. Second, a brief description of the different application contexts in which the formulations were evaluated is presented, with the objective of assessing the versatility and robustness of the proposed model across distinct problem settings. Finally, the results corresponding to the three sets of experiments are reported. For each case, numerical outcomes are presented and discussed, highlighting the comparative performance of the formulations and the practical implications of the observed differences.

4.1 Preliminaries

To illustrate the computational performance of the formulation proposed in this work, a series of numerical experiments are conducted on a self–scheduling UC problem context. In particular, the proposed approach is benchmarked against two alternative self–UC formulations, allowing for a comparative assessment. In order to validate our results, we solve each one of the instances using different state–of–the–art MILP solvers under identical computational settings, thereby ensuring a consistent and reliable evaluation of the proposed formulation.
As is well known, different MILP solvers employ distinct presolve strategies, heuristics, and cutting–plane techniques, which can significantly affect computational performance [92]. Therefore, the use of multiple solvers provides a more robust evaluation and yields results that are largely independent of the specific solution tool.
The computational performance of the proposed approach is evaluated by comparing the following three MILP formulations of the self–UC problem:

i) **M1:** This formulation follows the standard NF model wish was proposed in [12] and presented in section 2.2.2 with additional sides constraints corresponding to gradient between consecutive periods;

ii) **M2:** this formulation corresponds to the standard NF in **M1** and is strengthened by incorporating the set of valid inequalities proposed in Section (3.4.2), derived through the polyhedral analysis of the polytope $\mathcal{P}$ in (3.1);

iii) **M3:** this formulation refers to the *3bin* model proposed in [31], which explicitly captures commitment decisions, start–up, and shut–down states through three different binary variables.

Note that the three formulations differ solely in the algebraic representation of the constraints, while modeling the same underlying decision structure. Consequently, they characterize identical feasible sets in the original decision space and are therefore mathematically equivalent. In particular, for any given instance, all formulations admit identical optimal commitment decisions, generation levels, and objective function values.
Nevertheless, the formulations may differ substantially in their extended representations, leading to distinct polyhedral structures of the corresponding LP relaxations and, in turn, to different computational performance.
In this context, the computational time required to solve an instance remains one of the most important indicators to evaluate a given formulation. However, relying exclusively on solution time provides limited insight into the underlying reasons why one formulation outperforms another. In particular, *runtime* alone does not reveal whether insights about model compactness, tighter relaxations, or solver–specific effects. Consequently, complementary performance measures are necessary to better understand the structural advantages and limitations of competing formulations.
Accordingly, to evaluate the strength of a MILP formulation, metrics such as the integrality gap and the root–node gap are particularly relevant, as they provide information about the tightness of the LP relaxation and the expected difficulty of the branch–and–bound process.
For the optimality gap, lets define z_p as the primal objective bound and z_D as the dual objective bound, lower and upper bounds respectively in a maximization problem context, then the optimality gap, or just *gap*, is defined as $(z_D - z_P)/z_D\, 100\,\%$, the *gap* value is continuously updated during node exploration, until any of the stop criterion is met. For a given node in the B&B tree, the reported optimality gap quantifies the relative difference between the incumbent feasible solution and the best bound that the relaxation provides. In this context, the tightness of a MILP problem can be measured by the integrality gap, which is defined as the difference between the objective function value of the relaxed problem (z_{LP}) and that of the integer problem (z_{IP}) as $|z_{\text{IP}} - z_{\text{LP}}|/z_{\text{IP}}\, 100\,\%$, where given a maximization context, the relation $z_{\text{LP}} \geq z_{\text{IP}}$ is satisfied [93]. In particular, the closer the IP formulation is to the LP formulation, the better is the computational performance of the model. Another relevant metric for assessing the performance of a MILP formulation is the total number of simplex iterations performed during the B&B process. As is well known, the B&B algorithm solves a linear programming relaxation at each node of the search tree, typically using a simplex-based method. Consequently, the total number of simplex iterations reported by the solver provides a measure of the overall computational effort devoted to these LP relaxations and can serve as an indicator of the size and complexity of the search tree explored during the solution process.
All tests were carried out using a personal computer equipped with Intel Ultra 7, 2.2–`GHz` and 32 `GB` of `RAM`. The working environment and models are implemented using Python 3.10, with the mathematical programming language `AMPL`/`Gurobi` 11.00 as the MILP solver. We set an integrality gap tolerance equal to 10^{-6} and a time limit of 10^4 seconds, with all other solver parameters at their default values.
The following section presents the experimental setup and discusses the resulting solution times and performance indicators, providing insight into the practical applicability of the proposed approach.

4.2 Numerical experiments

The enhanced formulation for the operation of generating units through a feasible network path presented in Chapter 3 is evaluated in several contexts of increasing complexity. In particular, numerical experiments are conducted for the self–UC planning problem, the long–term power

system planning problem, and finally the self–scheduling of a virtual power plant under uncertainty. These settings allow us to assess the behavior of the formulation under different decision–making paradigms and information structures.
Within each of these planning contexts, a set of four numerical experiments is carried out to compare the performance of models **M1**, **M2**, and **M3**. The comparison is performed using the performance metrics introduced in the previous section, with the objective of evaluating both the computational efficiency and the structural quality of the formulations. In particular, this analysis seeks to assess the compactness and tightness of each model across the considered operational frameworks.

4.2.1 Self–UC

The self–UC problem refers to the decision–making process of a generating unit or a portfolio of generating unit that independently determines their commitment and dispatch levels in order to maximize its own profit given by the revenue from energy sales minus operating, fixed, and transition costs. The producer is assumed to be a price–taker[1], while accounting for technical constraints. In general terms, the self–UC problem for price–taker agents take the following form.

$$\max_{z \in \mathcal{Z}} \sum_{g \in \mathcal{G}} \sum_{t \in \mathcal{T}} \big(\pi_t \, q_{g,t}(z) - C_g(z) \big), \tag{4.1}$$

Where variables z collects all decision variables, $\mathcal{Z}$ is the feasible region induced by either the three–binary or the network–flow formulation, π_t correspond to the day–ahead market prices, which are exposed in Table 4.1, $q_{g,t}(z)$ denotes the induced power output, and C_g a general cost function for the respective generating unit.

Table 4.1: Energy price in \$/MWh [1].

$t = 1 \dots 12$	$\rightarrow$	13.0	7.2	4.6	3.3	3.9	5.9	9.8	15.0	22.1	31.3	33.2	24.8
$t = 13 \dots 24$	$\rightarrow$	19.5	16.3	14.3	13.7	15.0	17.6	20.2	29.3	49.5	53.4	30.0	20.2

Table 4.2 describes test–system used for the computational experiments of the present study–case, where technical information and cost coefficients of each unit are included. Regarding the technical information section, the minimum and maximum power outputs of the unit are denoted by parameters $\underline{P}$ and $\overline{P}$, respectively. Parameters RU and RD represent the ramp–up and ramp–down coefficients.

Table 4.2: Technical characteristics and cost coefficients of the portfolio of conventional fuel-based generating units considered in this analysis.

	Technical information								**Cost coefficients**		
NG	$\underline{\boldsymbol{P}}$ (MW)	$\overline{\boldsymbol{P}}$ (MW)	**RU** and **RD** (MW)	**SU** and **SD** (MW)	**TU** (h)	**TD** (h)	$\boldsymbol{p}_0$ (MW)	$\mathbf{UT}_0$ (h)	**CV** (\$/MWh)	**CF** (\$)	**CSD** (\$)
1	123	371	31	123	4	2	123	3	21	21	850
2	25	300	33	25	6	4	25	5	23	23	825
3	52	300	31	52	5	4	52	4	25	25	800
4	59	409	56	59	5	4	59	4	27	27	775
5	120	360	48	120	4	2	120	3	29	29	750
6	101	401	33	101	5	4	101	4	20	20	125
7	73	393	40	73	4	2	73	3	22	22	880
8	69	421	44	69	6	4	69	5	24	24	780

[1]A price taker is a market participant, typically an individual power plant or a small utility, that views market clearing prices as exogenous constants rather than variables it can influence [94].

In this section, the numerical experiments are conducted over different operational planning horizons of 128, 256, 365, and 512 days in order to assess the scalability and temporal robustness of the proposed formulation. For each time span, multiple generator portfolios are considered, ranging from reduced systems composed of subsets of one to four generating units to the complete portfolio of eight units reported in Table 4.2.

In this context, three experimental configurations are considered: *i*) In the first experiment, only generating unit 1 is included, *ii*) in the second experiment, the system comprises generating units 1 through 4, and *iii*) the third experiment considers the complete set of generating units. All experiments are conducted for each of the four planning horizons considered. We first examine the integrality gap for each study case, which constitutes a fundamental indicator of formulation quality. Since this metric remains relatively stable across the different time spans considered, we report the average integrality gap for each portfolio of generating units analyzed. Given that all three formulations yield the same optimal solution ($\boldsymbol{z_{IP}}$), one single value is reported. The variation across formulations arises only from the corresponding LP objective values.

Table 4.3 on the right reports the average integrality gap for each study case across the different time spans considered. As observed, the proposed formulation, referred as M2, exhibits a zero integrality gap between the LP relaxation and the IP formulation for all three study cases. In this context, the enhanced NF–based formulation for this study case coincides with the convex hull of the feasible region.

Table 4.3: Integrality gap comparison.

IG (%)	M1	M2	M3
I	46.243	0.000	24.901
II	80.865	0.000	25.684
III	78.401	0.000	27.462

Table 4.4 shows the branch&bound performance of the self–UC problem across four different metrics. The RootIGap metric measures the integrality gap in the root node of the B&B tree, prior to the branching process begins, which means that no cuts or solver–heuristics have been applied yet. The MIP time corresponds to the total solution time required by the solver to initialize the branch–and–bound procedure, including presolving, primal simplex, barrier methods, and other internal techniques used to compute the root-node objective value. The simplex iterations reported in the table correspond to the number of iterations performed by the simplex algorithm to solve the LP relaxations at each node of the tree.

Table 4.4: Computational performance of the different self–UC formulations across various configurations and time spans.

NC	**NP**	**RootIGap** (%)			**MIP time** (s)			**B&B nodes**			**Simplex iterations**		
		M1	M2	M3	M1	M2	M3	M1	M2	M3	M1	M2	M3
I	128	105	0	40.0	4.5	2.0	1.1	1	1	1	26592	20049	9684
	256	99.1	0	41.9	11.3	5.2	2.2	1	1	1	40060	38996	18726
	365	99.2	0	41.9	24.6	8.2	2.8	1	1	1	58149	56697	25928
	512	99.2	0	43.9	39.0	6.7	5.3	1	1	1	81661	75140	38146
II	128	4.4	0	84.7	10.9	8.7	3.1	1	1	1	28267	23584	8759
	256	4.4	0	32.9	40.1	22.6	8.5	1	1	1	55086	48395	17918
	365	4.3	0	33.0	56.3	31.2	14.7	1	1	1	77401	51159	26286
	512	4.3	0	33.1	170.1	30.9	20.8	1	1	1	102180	71796	37575
III	128	248	0	336	63.3	27.7	72.2	1	1	1	130873	24275	87927
	256	173	0	401	254.4	96.0	155.4	3939	1	4	279381	52002	178486
	365	173	0	1487	566.5	148.4	618.9	7425	1	18278	564600	46564	488307
	512	173	0	1491	1175.6	127.7	2045.6	8157	1	19444	751598	65330	616165

As shown in Table 4.4, the proposed model M2 exhibits considerably better performance than the

standard NF model, referred to as M1 for all the cases and time spans. However, despite M2 being tighter than model M3, model M3 demonstrates superior computational performance in terms of solution times for Cases I and II, with less effort to solve the model requiring fewer simplex iterations. In contrast, for larger instances such as Case III, model M2 exhibits a significantly better performance than model M3 across all considered metrics, indicating that the proposed formulation can be solved in less time and with lower computational effort than the other two models.

4.2.2 Power Systems Planning

In order to address the increasing demand for energy, power system expansion planning is a critical problem within the power systems sector. This decision–making problem enables system planners to assess future capacity requirements in both generation and transmission infrastructures, and to determine optimal investment and reinforcement strategies to ensure reliable and efficient system operation. In this context, power system expansion planning seeks to optimize infrastructure investments over a given planning horizon, accounting for long–term system needs and economic considerations [95], where the objective function consists, generally, of

$$\text{Objective function} = \text{Capital costs} + \text{Operation costs} \tag{4.2}$$

However, this problem inherently involves a trade–off between the level of temporal and operational detail used to represent system operation and the length of the planning horizon that can be considered within reasonable computational times. While a higher level of operational and temporal resolution allows for a more accurate representation of system dynamics–particularly in the presence of high shares of variable renewable energy sources—it significantly increases model complexity and computational cost. Conversely, coarser representations enable tractable long–term analyses but may lead to biased investment and dispatch decisions due to the simplified modeling of operational constraints and variability [96].

Power system expansion planning is inherently a multi-period optimization problem. For expository purposes, it can be conceptually decomposed into two interconnected decision-making levels. The first–level, commonly referred to as the upper-level problem, encompasses long-term investment decisions, including the selection of infrastructure assets and their associated capacity expansions. The second–level corresponds to the operational decision-making problem, which determines the optimal operation of the power system given the investment decisions established at the upper–level. In this context consider the following deterministic objective function for the power system planning [95]:

$$\min_{\substack{\boldsymbol{x}\in\mathcal{X},\\ \boldsymbol{y}\in\mathcal{Y}}} \sum_{i\in\Omega_{\mathcal{G}}} \tilde{I}_i^g\, x_i^g \sum_{k\in\Omega^{\mathcal{L}}} \tilde{I}_k^\ell\, x_k^\ell \;+\; \sigma \sum_{t\in\mathcal{T}} \left(\sum_g C_g^E\, y_g^E + \sum_d C_d^{LS}\, y_d^{LS} \right), \tag{4.3}$$

where the set $\boldsymbol{x}$ and $\boldsymbol{y}$ correspond to the first and second level decisions, respectively, while $\mathcal{X}$ and $\mathcal{Y}$ comprises the respective constraints of each of the stated decisions. In this context, the first and second terms of equation (4.3) represent the investment decisions in generation and transmission, respectively. The variables x_i^g and x_k^ℓ denote the corresponding investment decisions, while $\tilde{I}_i^g$ and $\tilde{I}_k^\ell$ represent the associated annualized investment costs of the candidate assets.

As depicted in equation (4.3), the operational cost term is multiplied by a scaling factor σ in order to ensure consistency between the operational costs over the planning horizon—given by the cardinality of the set $\mathcal{T}$ and the annualized investment costs represented by the first and second terms of the objective function. In this context, the computational tractability of the problem is directly related to the level of detail included in the set of operational decisions variables, denoted by $\mathcal{Y}$, as well as by the length of the planning horizon, represented by the set $\mathcal{T}$. In order to improve the tractability of the problem, several techniques have been proposed in the literature to reduce the number of periods in the planning horizon without compromising accuracy, using

the load duration curved (LDC) or working with typical days [97]. In parallel, other approaches focus on simplifying the set of operational decision variables $\mathcal{Y}$ by reducing the level of detail in the operational representation.
In this context, the formulation proposed in this study provides an accurate representation for addressing the objectives of power system planning regarding the tractability issue, particularly with respect to the lower–level problem associated with operational decisions.
To evaluate the performance of the proposed formulation within the context of the power system planning problem, we consider a single–bus generation expansion planning scenario in which the upper–level decisions are fixed, which means that the experiment consist of a long–term UC problem.

Table 4.5: Demand as a percentage (%) of the total system capacity, defined as the sum of the maximum power outputs of all generating units [2].

$t = 1 \dots 12$	$\rightarrow$	28%	27%	26%	24%	30%	38%	46%	54%	60%	68%	72%	78%
$t = 13 \dots 24$	$\rightarrow$	82%	75%	68%	74%	82%	78%	70%	62%	58%	50%	42%	31%

For this case, we set an integrality gap tolerance equal to 10^{-4} and a time limit equal to 10800 `s`, with all other solver parameters at their default values.
Table 4.6 present the performance of the proposed framework in solving the operational UC problem embedded into the power system planning problem using three different software tools, considering the three formulations, for different planning horizons and instance.

Table 4.6: Computational performance of the different self–UC formulations across various configurations and time spans.

NP	**NC**	**RootIGap** (%)			**MIP time** (s)			**B&B nodes**			**Simplex iterations**			**UserGap** 1e-2(%)		
		M1	M2	M3	M1	M2	M3	M1	M2	M3	M1	M2	M3	M1	M2	M3
128	1	21.7	19.7	24.2	152	46.8	1629.1	1	1	42169	287119	123780	1191948	0.96	0.98	1.00
	2	4.1	0.96	17.1	796.1	619.8	2012.6	5155	3153	45167	627892	396273	1207607	0.93	0.85	1.00
	3	23.3	20.1	70.8	1104.6	439.4	1267.1	2175	2226	2332321	759762	290823	62286802	0.89	0.73	2.05
	4	25.2	7.82	1.24	1333.2	313.5	10800	3241	2168	45484	853444	339400	1335000	1.00	0.96	1.00
	5	32.1	2.11	18.3	1391.6	434.9	950.8	3259	2216	51667	880492	326401	1290309	0.82	0.88	0.97
	6	12.2	8.34	17.5	152.8	18.2	2390.5	1	1	43369	178267	58902	1213709	0.00	0.67	0.77
	7	22.1	20.8	0.90	381.4	86.3	807.1	198	1	893356	351494	178890	20238978	0.98	0.98	3.44
	8	91.2	12.7	17.6	1501.9	511.7	10800	2332	2219	49469	872413	350528	1241125	0.55	0.82	1.00
	9	30.5	0.99	21.2	914.3	87.6	1646.2	2231	1	44815	792316	89074	1681587	0.59	0.33	0.98
	10	21.3	17.5	29.2	3105.4	721.2	2300.1	68269	6808	43356	5493881	734046	1610600	0.99	0.99	0.94
256	1	19.8	19.7	37.7	618.6	307.7	4665.5	2738	2243	44321	278210	344103	1478417	0.99	1.00	1.00
	2	2.83	0.96	18.0	2094.5	2420.9	2094.6	11700	3449	43644	715582	1077331	1569150	0.98	0.97	1.00
	3	27.3	20.1	70.9	1875.8	1648.0	10800	4674	4298	822201	704896	693950	21539055	0.97	0.94	1.85
	4	25.7	7.41	18.4	1891.4	1398.1	1890.7	6045	2880	43625	806196	646038	1607714	0.92	0.97	1.00
	5	30.2	1.50	18.1	3054.6	2353.2	3700.5	12652	3910	44083	1206948	1161358	1694220	0.89	0.92	1.00
	6	9.63	1.87	17.8	110.7	43.28	1796.3	1	1	44946	140383	120609	1452581	0.00	0.80	0.95
	7	27.6	20.4	0.90	609.2	476.7	10800	2527	2369	893356	301248	288136	20238978	0.97	0.99	3.44
	8	25.1	18.7	17.7	2332.9	2002.6	3635.7	7621	4288	44181	844026	954629	1357230	0.56	0.98	1.00
	9	4.44	0.79	1.92	6172.7	249.7	10800	42331	1	451022	2725246	177689	14389247	0.93	0.79	1.60
	10	21.3	17.5	28.9	10800	2393.4	6515.4	2126822	10623	30277	204320847	1226428	1510936	1.76	0.97	1.00
365	1	22.1	20.7	29.2	732.1	645.21	10800	2402	2118	243979	545257	342008	4315283	0.98	0.94	1.28
	2	2.98	0.96	18.6	3657.4	3669.5	8908.6	19570	7397	67048	1425782	1212351	2186578	0.75	0.96	1.00
	3	0.70	20.1	62.2	3700.0	3320.2	4135.8	11405	3386	44456	1233736	1309196	1585968	0.98	0.87	1.00
	4	21.4	7.12	18.0	4673.4	2451.9	10800	9154	4450	150241	1115638	1394957	4273708	0.97	0.76	2.15
	5	18.5	0.81	10.9	11884.7	5724.6	3041.2	6468	43456	43223	1739477	4836103	1644411	1.00	1.00	1.00
	6	9.64	1.78	19.0	155.6	69.8	7133.9	1	1	42043	197185	1649249	1396774	0.28	0.72	0.99
	7	5.67	19.24	0.91	1052.9	847.9	2456.6	2169	2306	41903	558028	405598	1396774	1.00	0.90	0.96
	8	1.67	0.15	17.9	6150.1	5309.2	10800		13073	340202	34358	1257679	7714555	0.94	0.90	3.75
	9	1.60	1.02	1.92	9008.7	509.4	6344.1	41432	1	44749	2653496	253913	1440348	0.89	0.69	0.98
	10	0.14	17.9	28.9	465.1	195.2	10800	1	1	156614	247196	217239	5318188	0.50	0.69	1.99

The RootIGap value in Table 4.6 is calculated as the relative difference between z_P and z_D prior to the branching begins, which means that no cuts or solver heuristics have been applied yet. A better gap at the root node can, in some cases, lead to better solution times, however this is not the general case, as each solver employs its own methods to tackle a problem, such as heuristics, cuts, and node presolve [98].
As is well known, the B&B algorithm solves an LP relaxation at every node of the search tree, typically using a simplex–based method. Therefore, the total number of simplex iterations reported by the solver reflects the overall effort spent solving these LP relaxations, and can be interpreted as an indicator of how deep or broad the search tree becomes during the solution process. As seen in Table 4.6, model M2 requires fewer simplex iterations than the other formulations in most instances. This reduction reflects the tightness of the LP relaxation of the model. In cases where the M1 formulation has fewer simplex iterations than M2, it is due to the latter using a barrier method alongside simplex method.
Figure 4.1 depicts the computational time required to solve each of the ten instances across the three considered planning horizons. It can be observed that model M2, corresponding to the enhanced NF formulation proposed in this work, tends to achieve lower computational times than models M1 and M3 across all instances and planning horizons considered. In particular, in terms of the median computational time, for the 128–day planning horizon, model M2 is solved approximately 2.7 times faster than model M1 and 4.9 times faster than model M3. For the 256–day planning horizon, model M2 is solved approximately 1.3 times faster than model M1 and 2.7 times faster than model M3. For the 365–day planning horizon, model M2 is solved approximately 2.2 times faster than model M1 and 4.8 times faster than model M3. It is also worth noting that formulation M2 exhibits less variability in CPU times across instances. In particular, the difference between the most computationally demanding instance and the easiest one is smaller than that observed for formulations M1 and M3, across all planning horizons. In this context, the reduced variability in CPU times observed for formulation M2 suggests that it provides more consistent computational performance across instances, being less sensitive to the complexity of individual instances compared to formulations M1 and M3.

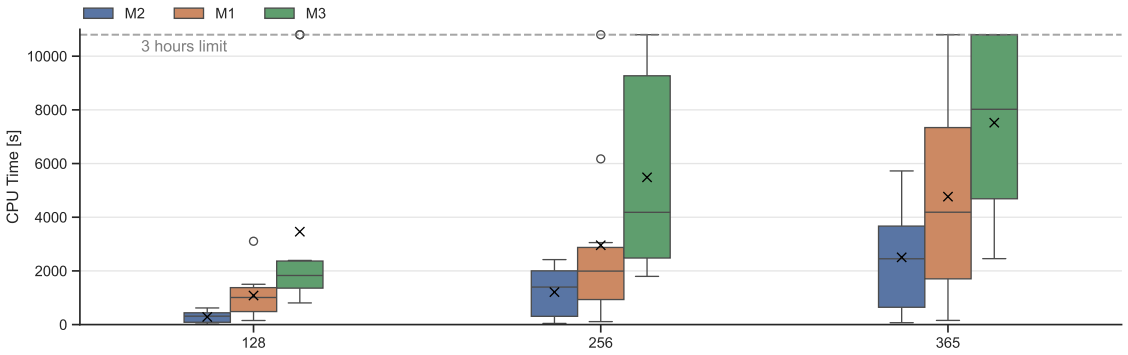
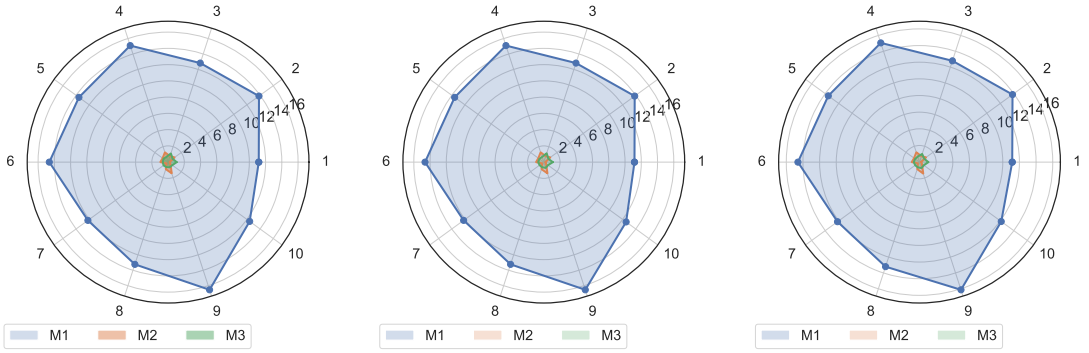


Figure 4.1: Computational time (in seconds) required per planning horizon for each of the ten analyzed instances, where the x–mark correspond to the mean CPU time value across all instances.

Figure 4.2 presents the integrality gap for all formulations, computed as the relative difference between the optimal integer solution and its corresponding LP relaxation. This metric provides an indication of the tightness of each formulation. As illustrated in the figure, formulation M2 exhibits a significantly smaller integrality gap compared to formulation M1, and a gap comparable to that of formulation M3, which is generally considered one of the tightest formulations for the UC problem. In this context, for the 128–day planning horizon, the average integrality gap of formulation M2 is approximately 17 times lower than that of formulation M1 and about 1.05 times higher than that of formulation M3, across all instances. For the 256–day planning horizon, the integrality gap of

formulation M2 is again approximately 17 times lower than that of formulation M1 and about 1.03 times higher than that of formulation M3. For the 356–day case, it is approximately 1.04 times higher than that of formulation M3, while remaining unchanged with respect to formulation M1. Overall, the integrality gap results indicate that formulation M2 exhibits a high degree of tightness, improving upon M1 and remaining consistently close to M3, which is widely regarded as one of the tightest formulations for the UC problem.



(a) Integrality gap for the 128–day planning horizon case.

(b) Integrality gap for the 256–day planning horizon case.

(c) Integrality gap for the 365–day planning horizon case.

Figure 4.2: Integrality gap (%) comparison among the three formulations evaluated over the ten analyzed instances.

Moreover, beyond studying the distance between the LP relaxation and the corresponding integer solution of a given formulation, it is essential to analyze the convergence behavior of the optimality gap. This gap is computed, as previously defined, as the difference between the best bound available in the branch-and-bound tree and the value of the incumbent solution, i.e., the best feasible integer solution identified during the execution of the B&B algorithm.

In this context, Figure 4.3 illustrates the convergence of the optimality gap for a specific instance evaluated over a 128–days planning horizon. As observed, formulation M3 is the first to produce a feasible incumbent solution, which enables the computation of the initial optimality gap earlier than the alternative models. However, despite this early feasibility detection, M3 exhibits pronounced difficulty in tightening the optimality gap, indicating a comparatively weak bound improvement and slower dual–primal convergence within the branch–and–bound process. In contrast, the proposed formulation demonstrates superior computational efficiency in closing the gap. After identifying a feasible solution, it achieves a significantly faster reduction of the optimality gap, ultimately converging to optimality in less computational time.

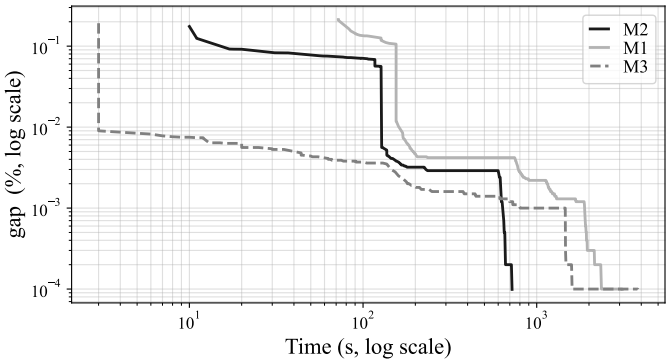


Figure 4.3: Convergence evolution for the optimality gap with logarithmic scale in both axis for instance n_1, evaluated over a 128–days planning horizon.

In this context, the fact that model M3 is able to compute an initial optimality gap earlier than models M1 and M2 can be attributed to the relative compactness of its formulation. A more compact model typically allows the solver to obtain an initial feasible solution and corresponding bound in shorter time. The figure further indicates that model M3 is more compact, as reflected

by the time taken to compute lower– and upper–bounds. However, model M2 exhibits a faster gap–closing behavior than model M3. This suggests that, although M2 may not be as compact as M3, its formulation achieves a more favorable trade–off between tightness and branch–and–bound efficiency. In particular, the strength of its relaxation and the resulting pruning of the search tree contribute to superior computational performance in terms of overall time to optimality.
These observations highlight that compactness alone does not determine computational efficiency; rather, the relation between formulation size, relaxation strength, and search–tree dynamics ultimately governs the performance of the solver, along with the solver own methods to solve a problem involving presolve techniques, heuristics, and cuts, that are typically solver–specific.
Finally, the second experiment, addressing the operational stage of the power system planning problem corresponding to a large–scale UC instance, demonstrates that the proposed model attains superior computational performance compared to both models M1 and M3. In particular, the enhanced formulation effectively strengthens the feasible region of the NF–based UC problem, as illustrated in Figure 4.2. This tightening yields stronger primal and dual bounds, and consequently, the overall computational performance improves, as evidenced by the reduced solution times.

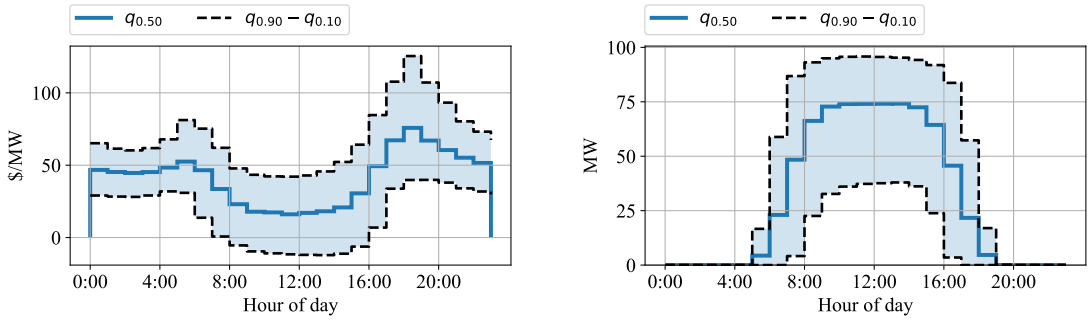
4.2.3 Self–scheduling of a virtual power plant

The self–scheduling of a Virtual Power Plant (VPP) refers to the optimization problem in which a VPP operator determines the optimal dispatch and market participation strategy of its aggregated resources for a given planning horizon. A Virtual Power Plant is an aggregation of heterogeneous distributed energy resources (DERs), such as renewable generation units (e.g., wind and photovoltaic systems), conventional generators, energy storage systems, and flexible loads. Through coordinated control and centralized optimization, the VPP operates as a single market entity in wholesale electricity markets [99].
This decision–making process is commonly formulated as a stochastic or robust optimization model when uncertainty is explicitly incorporated into the framework [100, 101, 102]. The objective function typically aims to maximize the expected profit of the VPP, while accounting for forecasted day–ahead electricity prices, technical constraints of generation units, including capacity limits, ramp–rate restrictions, and minimum up– and down–time requirements, operational constraints associated with energy storage systems, such as state–of–charge dynamics and efficiency losses, power balance conditions, and the uncertainty inherent in renewable generation and demand forecasts, among other relevant operational considerations. When accounting for the uncertainty, the problem can be generally expressed as:

$$\max_{\boldsymbol{x}\in\mathcal{X},\,\xi\in\Xi} \quad \boldsymbol{\rho}_{\xi}\left(\sum_{t\in\mathcal{T}}\pi_t p_t^E - C(\boldsymbol{x})\right) \tag{4.4}$$

Equation (4.4) represents the objective function of the self-scheduling problem of VPP. In this formulation, the decision vector $\boldsymbol{x} \in \mathcal{X}$ denotes the set of operational variables of the aggregated resources, while $\xi \in \Xi$ captures the uncertainty realization, such as renewable generation, demand, or price scenarios. The term $\boldsymbol{\rho}_{\xi}(\cdot)$ denotes a risk functional or expectation operator defined over the uncertainty set, allowing the VPP operator to incorporate risk preferences into the decision–making process. The first term in (4.4) corresponds to the revenues obtained by the VPP from its participation in the day–ahead energy market. Specifically, π_t denotes the forecasted electricity price at time period $t \in \mathcal{T}$, while p_t^E represents the amount of energy scheduled for exchange with the market during that period. The term $C(\boldsymbol{x})$ represents the operational costs associated with the dispatch of the aggregated resources. These costs may include generation costs of dispatchable units, as well as start-up and shut-down costs when unit commitment decisions are incorporated into the model.
In this context, as the last experiment, we propose comparing the performance of the formulation M1, M2, and M3 in the context of a self–scheduling planning problem of a VPP for large planning horizon, such as the considered in the previous experiments. We consider a VPP configuration

comprising two conventional generating units, a photovoltaic (PV) system, an energy storage system (ESS), and a flexible demand component, with the objective of maximizing profit over large planning horizon.
Both the day–ahead locational marginal prices (LMP), and the forecasted solar generation are obtained from the *"Open Access Same-time Information System"* (OASIS) from the California ISO [103]. In particular, LMPs are obtained for the generation node `TOT210S1_7_N002`, which is situated in proximity to the border between Inyo County, California, and Las Vegas, Nevada. The stochastic model utilizes a three-year of data ranging from 01/01/2023 to 12/31/2025, where each calendar year constitutes a distinct scenario. To ensure consistency across these scenarios, each one encompasses an analogous observation period; for instance, in a 128–day case study, the model evaluates the identical date range and seasonal window across all three years For the flexible demand, we consider a load profile similar to that of the last UC experiment. Regarding the ESS, and installed PV capacity, a configuration analogous to that proposed in [101] is adopted. Similarly, the photovoltaic (PV) generation profiles are derived from solar production data corresponding to the same zone as the LMPs. These profiles are subsequently scaled to ensure consistency with the capacity limits and modeling assumptions adopted in the framework described in the aforementioned study.
In this context, Figure 4.4a illustrates the average daily locational marginal price profile, together with the 90th and 10th percentile bands. Analogously, Figure 4.4b presents the average daily PV generation profile employed in the experiments. As previously discussed, this profile is appropriately scaled to a maximum installed capacity of 100 (`MW`).



(a) Average daily locational marginal price profile across all scenarios.

(b) Average daily PV generation profile across all scenarios.

Figure 4.4: Average daily profiles of LMPs and PV generation. The solid blue line represents the mean profile across all scenarios, while the upper and lower dotted black lines correspond to the 90th and 10th percentiles, respectively, based on historical data extracted from the CAISO market.

To assess the computational performance of the proposed formulation within the present framework, we consider a VPP configuration composed of one PV unit, one ESS, one flexible load, and a set of CPPs, with the number of CPPs varying from two to four.
Based on this structure, we conduct a series of computational experiments across different VPP configurations, parametrized by the number of CPPs, and across multiple planning horizons ranging from 128 to 365 days. This design allows us to evaluate the scalability of the formulation with respect to both the size of the generation portfolio and the temporal dimension of the problem.
Moreover, we analyze both the deterministic version of the VPP self–scheduling problem and its stochastic counterpart, thereby enabling a comprehensive assessment of the model's computational behavior under different uncertainty representations. In particular, for the deterministic case, we consider the centroid among the data set in the uncertainty set, while for the stochastic case, we incorporate the data from the years 2023 and 2024 in addition to 2025, with each year representing a distinct scenario in the stochastic programming framework. Additionally, for this last case, we

set an integrality gap tolerance equal to 10^{-6} and a time limit equal to 10800 s, with all other solver parameters at their default values.
In this context, Table 4.7 presents the computational performance of the B&B algorithm in solving the different instances of the VPP self–scheduling problem, for the deterministic formulation. The first column of the aforementioned table reports the number of planning periods considered in each instance, whereas the second column indicates the number of CPP units included in the VPP configuration.

Table 4.7: Computational performance comparison of the evaluated formulations across different configurations and time horizons in the **deterministic** setting, within a VPP framework comprising 2 to 4 generating units per period.

NP	**NG**	**RootIGap** (%)			**MIP time** (s)			**B&B nodes**			**Simplex iterations**		
		M1	M2	M3	M1	M2	M3	M1	M2	M3	M1	M2	M3
128	2	42.4	35.8	40.9	98.94	20.51	81.53	10094	1	5218	122863	102721	158553
	3	32.7	27.0	30.9	102.2	25.83	152.95	2501	1	7560	219546	78670	293687
	4	25.2	20.0	23.7	112.24	40.99	98.48	2788	1	4026	298508	135087	159140
256	2	26.8	23.1	19.2	102.40	45.31	130.79	1668	1	2893	248568	141951	144589
	3	22.1	18.7	17.0	218.51	72.88	286.96	1752	1	5938	187765	175823	260407
	4	17.6	14.3	26.5	268.97	97.78	373.41	1699	1	5938	243467	250671	355053
365	2	19.7	16.8	13.9	163.54	83.09	236.91	1739	1	6289	219833	165690	301715
	3	16.3	13.5	12.1	288.71	137.27	464.84	2493	1	12981	328504	265158	633499
	4	12.9	10.4	11.6	529.11	187.77	717.75	2783	1	7260	393155	399260	516303

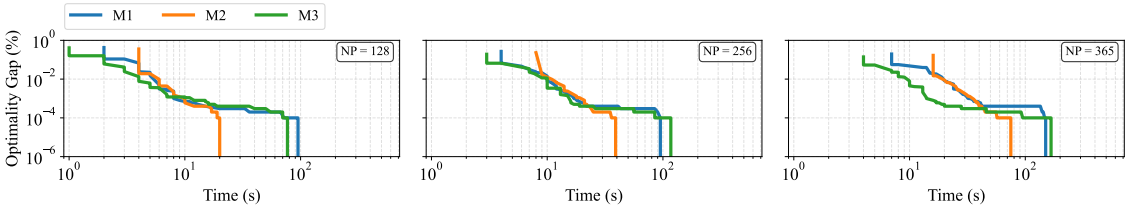
As reported in Table 4.7, the proposed formulation consistently exhibits superior computational performance across all evaluated metrics. In particular, the observed solution times are systematically lower than those obtained with the alternative formulations. Moreover, for all tested instances, optimality is achieved at the root node, with no need for branching. This behavior indicates that the continuous relaxation of the proposed model provides tighter bounds than those of the competing formulations, thereby substantially reducing the complexity of the branch-and-bound process. Additionally, the proposed formulation requires, in most instances, fewer simplex iterations compared to the other two models. This reduction suggests that less computational effort was required to solve the instances using the formulation M2.

Table 4.8: Computational performance comparison of the evaluated formulations across different configurations and time horizons in the **stochastic** setting, within a VPP framework comprising 2 to 4 generating units per period.

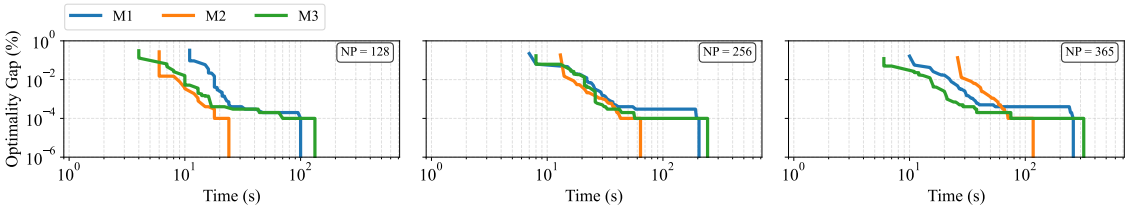
NP	**NG**	**RootIGap** (%)			**MIP time** (s)			**B&B nodes**			**Simplex iterations**		
		M1	M2	M3	M1	M2	M3	M1	M2	M3	M1	M2	M3
128	2	4.05	3.42	9.75	80.05	63.49	269.27	3160	1	20619	127640	98605	534396
	3	3.39	2.80	8.31	198.57	92.84	537.71	2858	1	22284	173604	65900	667768
	4	2.76	2.19	3.84	297.61	107.91	512.84	1927	1	11671	226735	122079	598459
256	2	9.23	3.27	5.57	669.68	157.16	865.51	8570	1	17942	757648	143540	792184
	3	11.0	2.73	3.06	1039.31	266.60	1960.24	4723	1	48786	1655303	184082	1539740
	4	8.84	5.54	7.02	1453.79	673.13	1438.01	3456	3	12371	2322847	650430	1123505
365	2	489	58.8	3.03	1097.62	484.91	1036.46	4005	3	14437	1671046	495399	877852
	3	9.72	6.10	3.04	1631.67	787.16	2793.75	3608	3	63613	2576874	829188	2646350
	4	7.75	4.77	5.67	2372.12	1250.79	2283.72	6264	3	30980	3407940	1201001	1943025

Similarly to the results reported in the previously discussed table, Table 4.8 presents the compu-

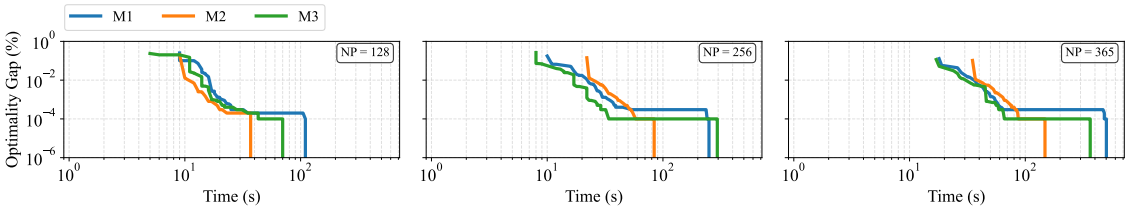
tational performance of the branch–and–bound algorithm when solving the stochastic counterpart of the considered models.
As indicated by the reported metrics, the stochastic formulation requires a higher computational effort than its deterministic counterpart. This increase is reflected in longer solution times, a larger number of explored nodes in the branch–and–bound tree, and, consequently, a greater number of simplex iterations. These results are consistent with the increased problem size and structural complexity induced by the incorporation of uncertainty.
Moreover, the results for this case align with those observed for the deterministic model, in that formulation M2 continues to exhibit the most favorable computational performance among the three formulations.
Furthermore, Figure 4.5 illustrates the evolution of the optimality gap over time for all instances considered within the deterministic model.



(a) Convergence of the optimality gap (log–log scale) for the instance with two CPP units, evaluated across all considered planning horizons.



(b) Convergence of the optimality gap (log–log scale) for the instance with three CPP units, evaluated across all considered planning horizons.

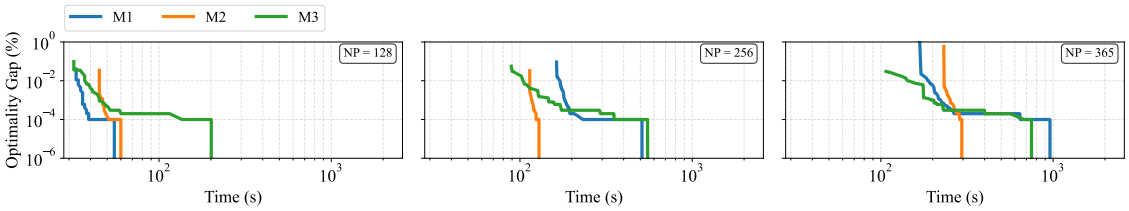


(c) Convergence of the optimality gap (log–log scale) for the instance with four CPP units, evaluated across all considered planning horizons..

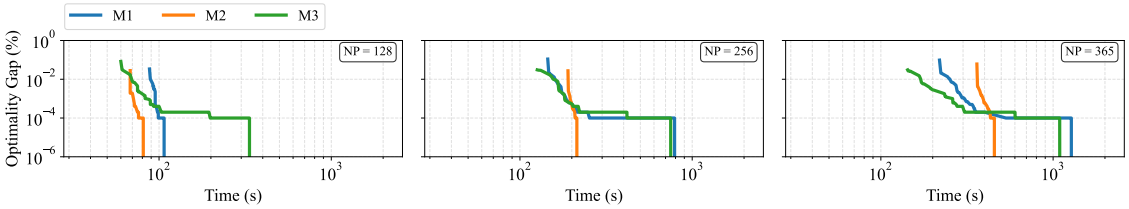
Figure 4.5: Convergence of the optimality gap for instances with 2, 3, and 4 CPP units, evaluated across all planning horizons for the **deterministic** model. Each subplot presents the gap evolution on a logarithmic scale for both axes, illustrating the computational behavior of the evaluated formulations as the number of generating units increases.

As can be noted from the observation of Figure 4.5, the formulation M3 is consistently the first to compute a an integrality gap among the two other instances, which is directly related to the compactness of the formulation, the feasible region is defined by a smaller number of hyperplanes. However, despite the fact that the formulation M1 is more efficient in computing a first gap, the formulation M2 requires less time to close the gap between the dual and the primal bounds,

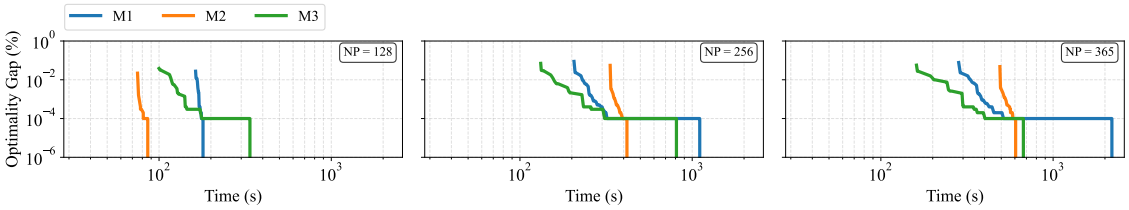
indicating a tighter relaxation and more efficient convergence within the branch-and-bound process. In this context, model M2 demonstrates superior scalability with respect to both the number of units and the length of the planning horizon, requiring comparatively lower computational effort than the other two formulations.
Similar to the previously discussed figure, Figure 4.6 illustrates the evolution of the optimality gap over time for all instances considered but within the stochastic model.



(a) Convergence of the optimality gap (log–log scale) for the instance with two CPP units, evaluated across all considered planning horizons.



(b) Convergence of the optimality gap (log–log scale) for the instance with three CPP units, evaluated across all considered planning horizons.

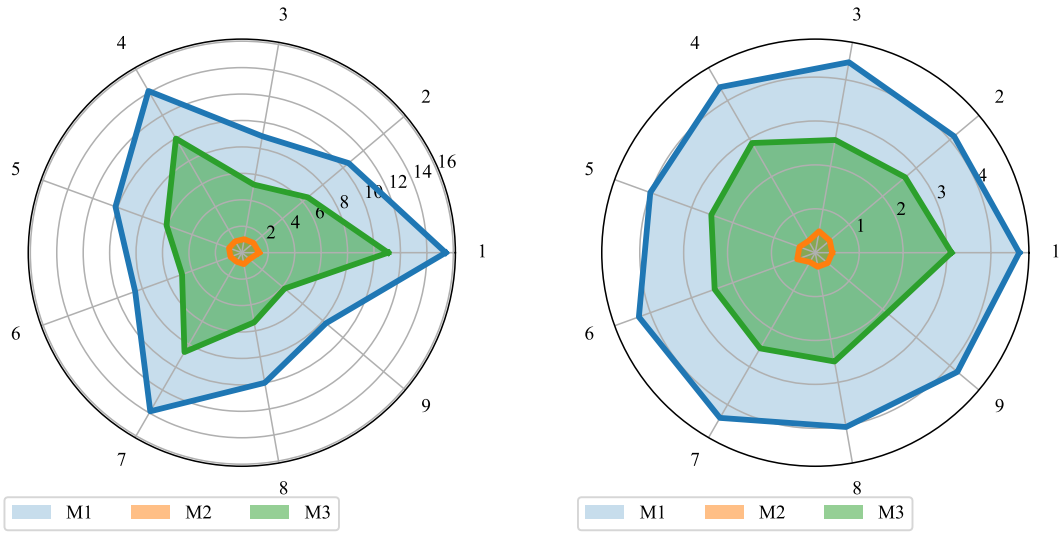


(c) Convergence of the optimality gap (log–log scale) for the instance with four CPP units, evaluated across all considered planning horizons..

Figure 4.6: Convergence of the optimality gap for instances with 2, 3, and 4 CPP units, evaluated across all planning horizons for the **stochastic** model. Each subplot presents the gap evolution on a logarithmic scale for both axes, illustrating the computational behavior of the evaluated formulations as the number of generating units increases.

As observed in Figure 4.6, the computational effort required to compute the initial integrality gap is higher than that reported for the deterministic model. Nevertheless, due to the relative compactness of formulation M3 compared to the other two formulations, it remains, as in the deterministic case, consistently the first to identify a gap. However, M3 exhibits limited scalability with respect to both the planning horizon and the number of units in the configuration, similar to formulation M1. In contrast, formulation M2 demonstrates efficient performance in closing the gap across all instances, highlighting its stronger computational behavior in larger or more complex problem settings. In this regard, and consistent with the observations from the deterministic analysis, formulation M2 exhibits improved scalability compared to its standard counterpart, namely formulation M1, as well as formulation M3, maintaining a more stable computational performance as problem size increases.

However, it is still relevant to discuss the time required by formulation M2 to compute the initial integrality gap. The comparatively longer time suggests that, relative to formulation M1, M2 is less compact in its polyhedral representation, as it relies on a larger number of inequalities to describe the feasible region. In this context, although the proposed facet–defining inequalities effectively tighten the feasible operational region of the generators, thereby strengthening the linear relaxation, this improvement is achieved at the expense of compactness.
Additionally, Figure 4.7 reports the integrality gap for each instance, computed as the relative difference between the optimal integer solution value and the optimal value of its linear relaxation. Specifically, Figure 4.7a displays the integrality gaps corresponding to the deterministic formulation, whereas Figure 4.7b presents those associated with the stochastic model. As can be observed, both figures exhibit a consistent pattern with respect to the relative magnitude of the integrality gaps across instances. In all cases, formulation M2 yields the tightest LP relaxation, as evidenced by the smallest integrality gaps. Formulation M3 ranks second in terms of relaxation strength, while formulation M1 consistently presents the weakest LP bounds.
This ordering indicates that the strengthening introduced in M2 effectively reduces the gap between the integer hull and its linear relaxation, thereby providing stronger lower bounds at the root node of the branch–and–bound tree.



(a) Integrality gap (%) for all instances within the **deterministic** formulation.

(b) Integrality gap (%) for all instances within the **stochastic** formulation.

Figure 4.7: Integrality gap comparison for the three formulations under study. Instances 1–3 correspond to the two-unit case evaluated over planning horizons of 128, 256, and 365 days, respectively. Instances 4–6 represent the three-unit case over the same planning horizons, while Instances 7–9 correspond to the four-unit case, likewise evaluated for 128, 256, and 365 days.

It is also relevant for the analysis to discuss the differences in the magnitudes of the integrality gap when comparing both models. In the context of pure–integer or mixed–integer programming problems, the instance data constitutes is relevant for the final computational performance of a given formulation.
Indeed, the numerical values defining the instance–such as cost coefficients, right–hand–side parameters, capacity limits, ramping bounds, and demand levels–directly shape the geometry of the feasible region and, consequently, can produce an impact the strength of the LP relaxation. Since the integrality gap is defined as the relative difference between the optimal integer solution and its continuous relaxation, its magnitude is influenced by how the data interacts with the constraint

system.
However, as Figure 4.7 suggests, the proposed formulation exhibits the lowest integrality gaps across all instances and for both the deterministic and stochastic models. This behavior indicates that the enhanced formulation consistently provides a tighter linear programming relaxation, independently of the underlying uncertainty representation. Consequently, the observed reduction in integrality gap provides strong evidence of the superior bounding quality of the proposed model, which helps explain its improved computational performance reported in the previous analysis.
Additionally, to further validate the robustness of the proposed results, a statistical analysis is conducted considering eight additional VPP configurations, characterized by different parameter settings of the CPP. For each configuration, the same experimental framework described previously is applied. In particular, the experiments involve scaling both the number of generating units and the length of the planning horizon allowing for a systematic assessment of computational performance and formulation strength under increasing problem dimensions.
Figure 4.8 shows the integrality gap for an additional set of 219 instances, solved using the stochastic model and generated across different VPP configurations and planning horizon lengths. In this regard, it is relevant to note that the same shape observed in the previous integrality gap comparison figure holds, namely, the proposed formulation consistently exhibits the lowest integrality gap across all analyzed instances and, therefore, corresponds to the tightest formulation among models M1 and M3.
Furthermore, according to the discussed figure, the proposed formulation consistently achieves the smallest integrality gap across all tested instances, thereby confirming the superior tightness of its linear relaxation when compared to the alternative models.

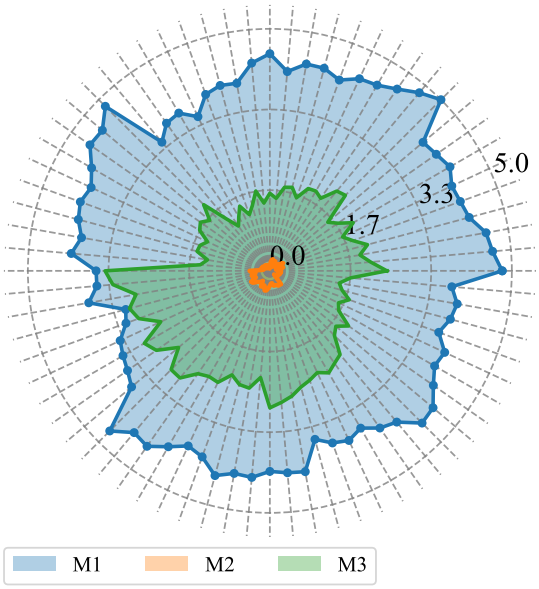


Figure 4.8: Integrality gap comparison for the three formulations under study for a large number of instances, encompassing different time periods and different VPP settings.

Additionally, although the primary objective of this work was to strengthen the LP relaxation of the NF-based formulation by incorporating tighter ramping constraints, which is exposed in the figure above, studying the improved computational times as a consequence of a tighter formulation is also relevant.
In this context, the speed–up heatmap shown in Figure 4.9 presents the computational time ratios of the instances considered in the statistical analysis, relative to formulation M2. The y-axis, labeled as instance–NG, represents the number of CPPs composing the VPP in each instance, whereas the x-axis, labeled as model/period, identifies both the formulation under study and the number of time periods defining the planning horizon.
Furthermore, the computational times of the proposed formulation are, in almost all cases, superior to those of the other two formulations within the studied context. In particular, model M2 achieves speed-ups of up to approximately 4.5 times in the best cases. In this regard, the results shown in Figure 4.9 are consistent with the findings of the previous analysis, demonstrating that the proposed formulation, besides being tighter, also achieves better computational times.

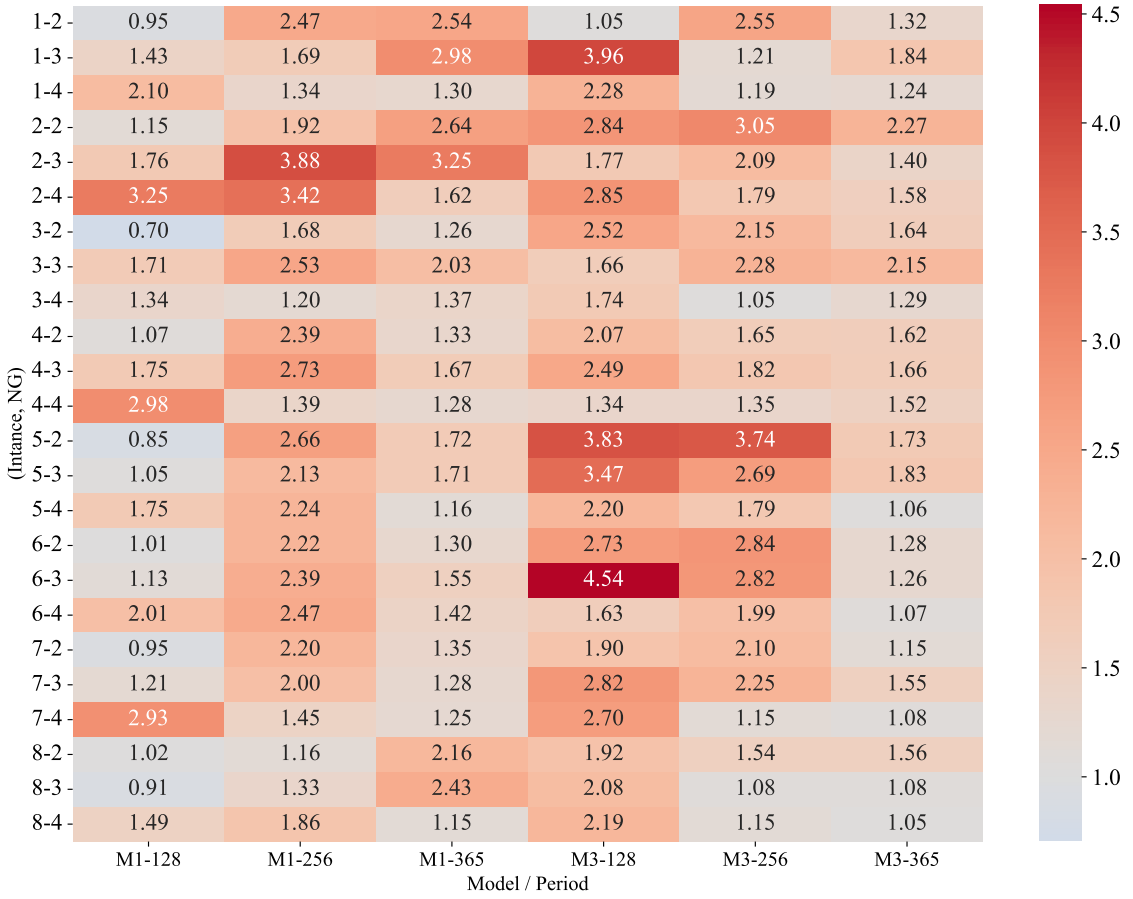


Figure 4.9: Speed–up heat–map exposing computational time ratios of the instances considered in the statistical analysis, relative to formulation M2. The y-axis (instance–NG) indicates the number of CPP units composing the VPP in each instance, while the x-axis (model/period) identifies the formulation and the corresponding number of time periods.

Chapter 5

Conclusions and future work

The present work has introduced an enhanced formulation for the NF–based operation of generation units subject to inter–temporal gradient limits between consecutive time periods. As previously discussed, the inclusion of ramping constraints negatively affects the geometry of the feasible region of the underlying formulation. In particular, these inter–temporal coupling constraints enlarge the set of fractional solutions in the continuous relaxation, thereby weakening the LP relaxation when compared to the corresponding model without ramping limits. In this context, the enhanced formulation proposed in this work aims precisely at mitigating this effect by providing a tighter representation of the feasible region in the presence of gradient limits.
In order to achieve a tighter representation of the formulation, it is essential to conduct a polyhedral study of the ramping polytope associated with the problem. In this context, the present work has carried out such a study, deriving facet–defining inequalities that strengthen the formulation. These inequalities are not only applicable within the specific context of energy systems operation but can also be extended to a broader class of problems involving networked systems coupled across time, such as: production and inventory planning in manufacturing systems, transportation and logistics networks, water or gas distribution networks, telecommunication or data networks, among others. In this regard, a polyhedral study was conducted to derive facet–defining inequalities aimed at strengthening the feasible region for the general structure composed by a node with entering and leaving edges. This analysis was carried out using the software PORTA, which allows the user to explore the complex geometry of a high–dimensional polyhedron using only its description in terms of extreme points. Through the analysis of a large set of instances, candidate facet–defining inequalities for the ramping polyhedron were first identified. Subsequently, these candidates were rigorously validated following the scientific method: hypotheses regarding their facet–defining nature were formulated, tested against multiple extreme points and instances, and only inequalities satisfying the necessary conditions for being facet-defining were retained.
Subsequently, a formulation for NF–based operation of generating units subject to ramping constraint was proposed, incorporating the facet–defining inequalities for the ramping polytope derived from the preceding polyhedral analysis. Following the development of this enhanced formulation, a series of computational experiments were conducted to evaluate its performance. The experiments were carried out across three different contexts: the self–commitment of generating units problem in the day–ahead market, long–term power system planning, and finally the self–scheduling problem of a virtual power plant also in the day–ahead market, including both the deterministic formulation along with its deterministic counterpart. The proposed formulation was compared in this three different context against the standard NF formulation, referred in the analysis as M1, and the *3bin* formulations, referred as M3, in terms of relevant metrics such as: relaxation tightness, integrality gap, and overall computational efficiency.
Consequently, the principal conclusions and key observations emerging from the present work are summarized as follows:

1. The facet–defining inequalities derived through the polyhedral analysis of the ramping polytope, effectively reduces the distance between the LP relaxation and the IG formulation, obtaining a tighter region compared to the standard counterpart. This improvement translates into consistently reduced computational times across all experimental contexts considered.

2. Given the tighter region, the proposed model demonstrates improved scalability in terms of the planning horizon, and the number of generating units, consistently achieving reduced computational times and lower computational burden across all studied cases compared to the other formulations. This property makes the model particularly suitable for long-term power system planning problems, such as investment analysis, expansion planning, or multi–year operational studies.

3. The proposed formulation exhibits superior performance compared to the other models, particularly in the self–scheduling cases, namely the first and third experiment. This outcome is consistent with the underlying operational behavior: in these cases, units aim to maximize profits by increasing their power output as much as possible, which is constrained by the ramping limits. The facet-defining inequalities in the proposed model effectively tighten the relaxation under these conditions, allowing the solver to handle these constraints more efficiently.

In light of the results obtained and the theoretical developments presented throughout this thesis, several research directions naturally emerge. First, future work should aim to identify a more selective set of nodes at which to apply the additional facet-defining inequalities derived in this study, with the aim of reducing the size of the resulting enhanced formulation while preserving the improvements in relaxation tightness. Second, future work will investigate the potential extension and applicability of the polyhedral results obtained in this thesis to other classes of network optimization problems. In particular, it will explore whether the facet–defining inequalities and strengthening techniques developed for the ramping polytope can be adapted to alternative time–coupled or flow–based formulations.
Finally, the present work highlights the relevant trade–off between tightness and compactness in MILP formulations in the context of NF–based operation of generating units with ramping constraints. The enhanced formulation proposed in this work, strengthened by facet–defining inequalities derived from the polyhedral analysis of the ramping polytope, prioritizes tightness, as the carried experiments suggests. While the addition of these inequalities increases the number of constraints and, consequently the size of the formulation, the computational experiments demonstrate that the trade–off is favorable: the tighter relaxation accelerates the branch–and–bound process and reduces overall solution time, even for large instances and extended planning horizons.

Bibliography

[1] Germán Morales-España, Claudio Gentile, and Andres Ramos. Tight mip formulations of the power-based unit commitment problem. *OR Spectrum*, 37(4):929–950, Oct 2015.

[2] 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):39–46, Feb 2012.

[3] Miguel F Anjos and Antonio J Conejo. Unit commitment in electric energy systems. *Foundations and trends in electric energy systems*, 1(4):220–310, 2017.

[4] Michael Jünger, Thomas M Liebling, Denis Naddef, George L Nemhauser, William R Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A Wolsey. *50 Years of integer programming 1958-2008: From the early years to the state-of-the-art*. Springer Science & Business Media, 2009.

[5] Ricardo M Lima and Augusto Q Novais. Symmetry breaking in milp formulations for unit commitment problems. *Computers & Chemical Engineering*, 85:162–176, 2016.

[6] James Ostrowski, Miguel F Anjos, and Anthony Vannelli. Modified orbital branching for structured symmetry with an application to unit commitment. *Mathematical Programming*, 150(1):99–129, 2015.

[7] François Margot. *Symmetry in Integer Linear Programming*, pages 647–686. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[8] Pelin Damcı-Kurt, Simge Küçükyavuz, Deepak Rajan, and Alper Atamtürk. A polyhedral study of production ramping. *Mathematical Programming*, 158(1):175–205, 2016.

[9] Pascale Bendotti, Pierre Fouilhoux, and Rottner. On the complexity of the unit commitment problem. *Annals of Operations Research*, 274:119–130, 2019.

[10] Robert E Bixby, Mary Fenelon, Zonghao Gu, Ed Rothberg, and Roland Wunderling. Mixed-integer programming: A progress report. In *The sharpest cut: the impact of Manfred Padberg and his work*, pages 309–325. SIAM, 2004.

[11] Robert E Bixby. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, 2012:107–121, 2012.

[12] Xiaohong Guan, Peter B Luh, Houzhong Yan, and JA Amalfi. An optimization-based method for unit commitment. *International Journal of Electrical Power & Energy Systems*, 14(1):9–17, 1992.

[13] R. Fourer, D. M. Gay, and B. Kernighan. Algorithms and model formulations in mathematical programming. In Stein W. Wallace, editor, *Algorithms and Model Formulations in Mathematical Programming*, chapter AMPL: A Mathematical Programming Language, pages 150–151. Springer-Verlag, Berlin, Heidelberg, 1989.

[14] Gurobi Optimization Inc. Gurobi Optimizer Reference Manual, 2014. http://www.gurobi.com.

[15] IBM Corporation. *IBM ILOG CPLEX Optimization Studio*. Armonk, NY, 2024. Version 22.1.1.

[16] IBM ILOG CPLEX Optimizer. urlhttp://www-01.ibm.com/software/integration/optimization/cplex-optimizer/, Last 2010.

[17] Richard H Byrd, Jorge Nocedal, and Richard A Waltz. Knitro: An integrated package for nonlinear optimization. In *Large-scale nonlinear optimization*, pages 35–59. Springer, 2006.

[18] Fair Isaac Corporation. *FICO Xpress Optimization*. San Jose, CA, 2024. Version 9.4.

[19] Christian PRINS. Applications of optimization with xpress-mp. *contract*, 2002.

[20] MOSEK ApS. *The MOSEK Optimizer API for Python*. Copenhagen, Denmark, 2024. Version 10.2.

[21] Thomas Christof, Andreas Löbel, and M Stoer. Porta-polyhedron representation transformation algorithm. *Software package, available for download at https://porta.zib.de/*, 1997.

[22] G. Morales-Espa na, J. M. Latorre, and A. Ramos. Tight and compact milp formulation for the thermal unit commitment problem. *IEEE Transactions on Power Systems*, 28(4):4897–4908, Nov 2013.

[23] Dimitri Bertsekas. *Network optimization: continuous and discrete models*, volume 8. Athena Scientific, 1998.

[24] Gary R Waissi. Network flows: Theory, algorithms, and applications, 1994.

[25] Michele Conforti, Marco Di Summa, Friedrich Eisenbrand, and Laurence A Wolsey. Network formulations of mixed-integer programs. *Mathematics of Operations Research*, 34(1):194–209, 2009.

[26] Bingdong Li, Jeff Springer, George Bebis, and Mehmet Hadi Gunes. A survey of network flow applications. *Journal of Network and Computer Applications*, 36(2):567–581, 2013.

[27] Robert Fourer, David M. Gay, and Brian W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Thomson/Brooks/Cole, Pacific Grove, CA, 2nd edition, 2003.

[28] Ashish Goel. *Introduction to Optimization (MS&E 111/ENGR 62), Chapter 5: Convexity, BFS, and Optimality*. Stanford University, Department of Management Science and Engineering, 2007. Course Lecture Notes.

[29] Amgad Madkour, Walid G Aref, Faizan Ur Rehman, Mohamed Abdur Rahman, and Saleh Basalamah. A survey of shortest-path algorithms. *arXiv preprint arXiv:1705.02044*, 2017.

[30] Deepak Rajan, Samer Takriti, et al. Minimum up/down polytopes of the unit commitment problem with start-up costs. *IBM Res. Rep*, 23628:1–14, 2005.

[31] M. Carrion and J. M. Arroyo. A computationally efficient mixed–integer linear formulation for the thermal unit commitment problem. *IEEE Trans. Power Systems*, 21(3):1371–1378, Aug 2006.

[32] Antonio Frangioni and Claudio Gentile. Solving nonlinear single-unit commitment problems with ramping constraints. *Operations Research*, 54(4):767–775, 2006.

[33] A. Frangioni, C. Gentile, and F. Lacalandra. Tighter approximated milp formulations for unit commitment problems. *IEEE Transactions on Power Systems*, 24(1):105–113, Feb 2009.

[34] Ben Knueven, Jim Ostrowski, and Jianhui Wang. The ramping polytope and cut generation for the unit commitment problem. *INFORMS Journal on Computing*, 30(4):739–749, 2018.

[35] Bernard Knueven, James Ostrowski, and Jean-Paul Watson. On mixed-integer programming formulations for the unit commitment problem. *INFORMS Journal on Computing*, 32(4):857–876, 2020.

[36] Narayana Prasad Padhy. Unit commitment-a bibliographical survey. *IEEE Transactions on power systems*, 19(2):1196–1205, 2004.

[37] K. Hara, M. Kimura, and N. Honda. A method for planning economic unit commitment and maintenance of thermal power systems. *IEEE Transactions on Power Apparatus and Systems*, PAS-85(5):427–436, May 1966.

[38] F. N. Lee. A fuel-constrained unit commitment method. *IEEE Transactions on Power Systems*, 4(3):1208–1218, Aug 1989.

[39] RM Burns. Optimization of priority lists for a unit commitment program. In *Proc. IEEE Power Eng. Soc. Summer Meeting, 1975*, 1975.

[40] Fred N Lee. Short-term thermal unit commitment-a new method. *IEEE Transactions on Power Systems*, 3(2):421–428, 2002.

[41] W. L. Snyder, H. D. Powell, and J. C. Rayburn. Dynamic programming approach to unit commitment. *IEEE Transactions on Power Systems*, 2(2):339–348, May 1987.

[42] C. k. Pang, G. B. Sheble, and F. Albuyeh. Evaluation of dynamic programming based methods and multiple area representation for thermal unit commitments. *IEEE Transactions on Power Apparatus and Systems*, PAS-100(3):1212–1218, March 1981.

[43] Gregory S Lauer, NR Sandell, DP Bertsekas, and Thomas A Posbergh. Solution of large-scale optimal unit commitment problems. *IEEE Transactions on Power Apparatus and Systems*, (1):79–86, 2007.

[44] Arthur I Cohen and Miki Yoshimura. A branch-and-bound algorithm for unit commitment. *IEEE Transactions on power apparatus and Systems*, (2):444–451, 2007.

[45] Nan Yang, Zhenqiang Dong, Lei Wu, Lei Zhang, Xun Shen, Daojun Chen, Binxin Zhu, and Yikui Liu. A comprehensive review of security-constrained unit commitment. *Journal of Modern Power Systems and Clean Energy*, 10(3):562–576, 2021.

[46] Xinming Lin, Z Jason Hou, Huiying Ren, and Feng Pan. Approximate mixed-integer programming solution with machine learning technique and linear programming relaxation. In *2019 3rd International Conference on Smart Grid and Smart Cities (ICSGSC)*, pages 101–107. IEEE, 2019.

[47] Álinson S Xavier, Feng Qiu, and Shabbir Ahmed. Learning to solve large-scale security-constrained unit commitment problems. *INFORMS Journal on Computing*, 33(2):739–756, 2021.

[48] Fouad Hasan, Amin Kargarian, and Ali Mohammadi. A survey on applications of machine learning for optimal power flow. In *2020 IEEE Texas Power and Energy Conference (TPEC)*, pages 1–6, 2020.

[49] Yafei Yang and Lei Wu. Machine learning approaches to the unit commitment problem: Current trends, emerging challenges, and new strategies. *The Electricity Journal*, 34(1):106889, 2021.

[50] Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid Von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, et al. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*, 2020.

[51] Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization*. John Wiley & Sons, 2014.

[52] G. B. Sheble and G. N. Fahd. Unit commitment literature synopsis. *IEEE Transactions on Power Systems*, 9(1):128–135, Feb 1994.

[53] L. L. Garver. Power generation scheduling by integer programming-development of theory. *Transactions of the American Institute of Electrical Engineers. Part III: Power Apparatus and Systems*, 81(3):730–734, 1962.

[54] Peter Malkin and L Wolsey. Minimum runtime and stoptime polyhedra. *Report, CORE, Université catholique de Louvain*, 2003.

[55] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[56] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, USA, 1988.

[57] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[58] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, San Francisco, 1979.

[59] Komei Fukuda et al. Frequently asked questions in polyhedral computation. *ETH, Zurich, Switzerland*, 85:10–35, 2004.

[60] Günter M Ziegler. *Lectures on polytopes*, volume 152. Springer Science & Business Media, 2012.

[61] Laurence A Wolsey. *Integer programming*. John Wiley & Sons, 2020.

[62] Yves Pochet and Laurence A Wolsey. *Production planning by mixed integer programming*, volume 149. Springer, 2006.

[63] A. Klar. Cutting planes in mixed integer programming. Master's thesis, Technische Universität Berlin, Berlin, Germany, 2006.

[64] Ed Rothberg. The CPLEX library: Presolve and cutting planes. Slides from the 4th Max-Planck Summer School: ADFOCS 2003, 2003. Presented at Saarbrücken, Germany, September 8-12.

[65] Tobias Achterberg. Constraint integer programming. *PhD. Thesis, Technical University of Berlin*, 2007.

[66] E. Balas. Facets of the knapsack polytope. *Mathematical Programming*, 8:146 – 164, 1975.

[67] L. A. Wolsey. Facets of linear inequalities in 0-1 variables. *Mathematical Programming*, 8:165 – 178, 1975.

[68] Laurence A Wolsey. Valid inequalities for 0–1 knapsacks and mips with generalised upper bound constraints. *Discrete Applied Mathematics*, 29(2-3):251–261, 1990.

[69] T. Van Roy and L. Wolsey. Valid inequalities for mixed 0-1 programs. *Discrete Applied Mathematics*, 14:199 – 213, 1986.

[70] Z. Gu, G. Nemhauser, and M. Savelsbergh. Lifted flow cover inequalities for mixed 0-1 integer programs. *Mathematical Programming*, 85:439 – 468, 1999.

[71] Hugues Marchand and Laurence A Wolsey. The 0-1 knapsack problem with a single continuous variable. *Mathematical Programming*, 85:15–33, 1999.

[72] M. W. Padberg, T. J. van Roy, and L. A. Wolsey. Valid linear inequalities for fixed charge problems. *Operations Research*, 33(4):pp. 842–861, 1985.

[73] I. R. de Farias and M. Zhao. A polyhedral study of the semi-continuous knapsack problem mathematical programming. *Mathematical Programming*, pages 1 – 35, 2012.

[74] Gustavo Angulo, Shabbir Ahmed, and Santanu S Dey. Semi-continuous network flow problems. *Mathematical Programming*, 145:565–599, 2014.

[75] Alejandro Angulo, Daniel Espinoza, and Rodrigo Palma. Sequence independent lifting for mixed knapsack problems with gub constraints. *Mathematical Programming*, 154:55–80, 2015.

[76] Ralph E Gomory. *Outline of an algorithm for integer solutions to linear programs and an algorithm for the mixed integer problem.* Springer, 2010.

[77] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[78] Eugene L Lawler and David E Wood. Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719, 1966.

[79] David R Morrison, Sheldon H Jacobson, Jason J Sauppe, and Edward C Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, 2016.

[80] Tobias Achterberg, Thorsten Koch, and Alexander Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2005.

[81] Louis Caccetta. *Branch and Cut Methods for Mixed Integer Linear Programming Problems*, pages 21–44. Springer US, Boston, MA, 2000.

[82] E Robert Bixby, Mary Fenelon, Zonghao Gu, Ed Rothberg, and Roland Wunderling. Mip: Theory and practice—closing the gap. In *IFIP Conference on System Modeling and Optimization*, pages 19–49. Springer, 1999.

[83] Tobias Achterberg and Roland Wunderling. Mixed integer programming: Analyzing 12 years of progress. In *Facets of combinatorial optimization: Festschrift for martin grötschel*, pages 449–481. Springer, 2013.

[84] Laurence A Wolsey. Strong formulations for mixed integer programs: valid inequalities and extended formulations. *Mathematical programming*, 97(1):423–447, 2003.

[85] Tobias Achterberg, Robert E Bixby, Zonghao Gu, Edward Rothberg, and Dieter Weninger. Presolve reductions in mixed integer programming. *INFORMS Journal on Computing*, 32(2):473–506, 2020.

[86] Alexander Schrijver. *Theory of linear and integer programming.* John Wiley & Sons, 1998.

[87] Sheng Liu and Deepak Rajan. A study of three-period ramp-up polytope. Technical report, Technical Report, 2015.

[88] Gunar E. Liepins. Fourier-motzkin elimination for mixed systems. Technical Report ORNL/TM-9669, Oak Ridge National Laboratory, Oak Ridge, TN, 1985.

[89] Michel L Balinski. An algorithm for finding all vertices of convex polyhedral sets. *Journal of the Society for Industrial and Applied Mathematics*, 9(1):72–88, 1961.

[90] TH Matheiss and David S Rubin. A survey and comparison of methods for finding all vertices of convex polyhedral sets. *Mathematics of operations research*, 5(2):167–185, 1980.

[91] Marc Pfetsch. Polyhedral computations: An introduction to Porta and polymake. Lecture and Exercises, Block Course: "Combinatorial Optimization at Work", October 2005.

[92] Jan Kronqvist, David E Bernal, Andreas Lundell, and Ignacio E Grossmann. A review and comparison of solvers for convex minlp. *Optimization and Engineering*, 20(2):397–455, 2019.

[93] H Paul Williams. *Model building in mathematical programming*. John Wiley & Sons, 2013.

[94] Juan Arteaga and Hamidreza Zareipour. A price-maker/price-taker model for the operation of battery storage systems in electricity markets. *IEEE Transactions on Smart Grid*, 10(6):6912–6920, 2019.

[95] Antonio J. Conejo, Luis Baringo Morales, S. Jalal Kazempour, and Afzal S. Siddiqui. *Investment in Electricity Generation and Transmission: Decision Making under Uncertainty*. Springer, Cham, 2016.

[96] Kris Poncelet, Erik Delarue, Daan Six, Jan Duerinck, and William D'haeseleer. Impact of the level of temporal and operational detail in energy-system planning models. *Applied Energy*, 162:631–643, 2016.

[97] Hossein Seifi and M Sadegh Sepasian. Electric power system planning: issues, algorithms and solutions. *Power Systems*, 49, 2011.

[98] Diego A Tejada-Arango, Sara Lumbreras, Pedro Sánchez-Martín, and Andres Ramos. Which unit-commitment formulation is best? a comparison framework. *IEEE Transactions on Power Systems*, 35(4):2926–2936, 2019.

[99] Natalia Naval and Jose M Yusta. Virtual power plant models and electricity markets-a review. *Renewable and Sustainable Energy Reviews*, 149:111393, 2021.

[100] Ana Baringo and Luis Baringo. A stochastic adaptive robust optimization approach for the offering strategy of a virtual power plant. *IEEE transactions on power systems*, 32(5):3492–3504, 2016.

[101] Ana Baringo, Luis Baringo, and José M Arroyo. Day-ahead self-scheduling of a virtual power plant in energy and reserve electricity markets under uncertainty. *IEEE Transactions on Power Systems*, 34(3):1881–1894, 2018.

[102] Yunfan Zhang, Feng Liu, Zhaojian Wang, Yifan Su, Weisheng Wang, and Shuanglei Feng. Robust scheduling of virtual power plant under exogenous and endogenous uncertainties. *IEEE Transactions on Power Systems*, 37(2):1311–1325, 2021.

[103] California Independent System Operator. Open Access Same-time Information System (OASIS). `https://oasis.caiso.com/`, 2025. Accessed: 2025-02-12.